GitHub Link

https://github.com/zonggen/csc598lho-project

Prepara Data

In [1]:
```
!rm -rf event.db && wget https://raw.githubusercontent.com/YifanDengWHU/DDIM
!ls -l
```

```
--2024-04-10 04:53:40--  https://raw.githubusercontent.com/YifanDengWHU/DDIM
DL/master/event.db
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.1
08.133, 185.199.109.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.
108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 30580736 (29M) [application/octet-stream]
Saving to: ‘event.db’

event.db            100%[===================>]  29.16M   106MB/s    in 0.3s

2024-04-10 04:53:41 (106 MB/s) - ‘event.db’ saved [30580736/30580736]

total 29868
-rw-r--r-- 1 root root 30580736 Apr 10 04:53 event.db
drwxr-xr-x 1 root root     4096 Apr  8 13:27 sample_data
```

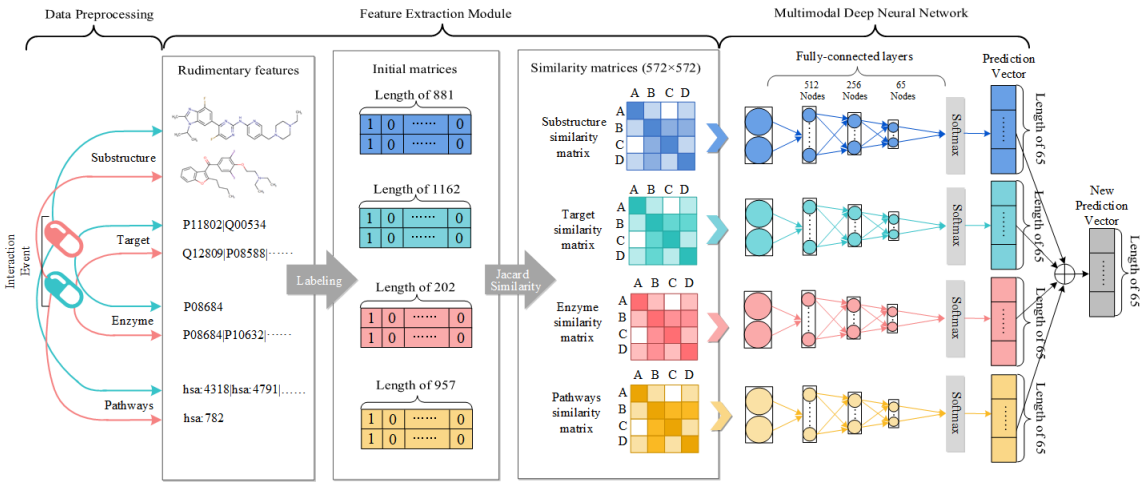
Introduction

The motivation behind the study lies in the critical need to predict drug-drug interactions (DDIs). DDIs occur when two or more drugs interact in a way that alters the effect of one or all of the drugs. These interactions can lead to unexpected side effects, reduce the effectiveness of the drugs, or increase the action of the drugs, which can be harmful or even deadly. The study aims to address these challenges by developing a deep learning framework that can accurately predict DDIs, contributing to safer and more effective treatment strategies. The ultimate goal is to improve patient outcomes and advance personalized medicine.

This study obtains the DDI data from DrugBank, and applies NLP techniques to classify DDI-associated events into 65 types according to their descriptions' syntax, and compiles a dataset of 572 drugs, 74 528 interactions and 65 types of DDI-associated events. A multimodal deep learning framework named DDIMDL that combines diverse drug features with deep learning is presented for the DDI event prediction. Evaluated using 5-fold cross validations, DDIMDL outperforms the existing DDI event prediction

method and baseline methods. The case studies are also performed to identify the DDI events not included in our dataset, and several DDI-associated events, such as the event caused by the interaction between Dextroamphetamine and Fenfluramine, are successfully found out.

A convolutional neural network based deep learning framework called DDIMDL is proposed, utilizing multiple drug features such as chemical substructures, targets, enzymes, and pathways to predict DDI events, using the DrugBank database:



Scope of Reproducibility:

I will evalutate and reproduce the model performance between the DNN model from the paper as well as other common classifier models, including RandomForestClassifier, GradientBoostingClassifier, SVM, GradientBoostintgClassifier, KNearestNeighborClassifier and LogisticRegression. The ablations are done through replacing the proposed DDIMDL model with other common classifiers.

1. Hypothesis 1: Performance measurement with DNN / DDIMDL
2. Hypothesis 2: Performance measurement with RF
3. Hypothesis 3: Performance measurement with GBDT
4. Hypothesis 4: Performance measurement with SVM
5. Hypothesis 5: Performance measurement with KNN
6. Hypothesis 6: Performance measurement with LogisticRegression

Methodology

Environment Setup

In [2]:
```
!pip install Keras numpy pandas scikit-learn tensorflow_decision_forests mat
```

```
Requirement already satisfied: Keras in /usr/local/lib/python3.10/dist-packa
ges (2.15.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packa
ges (1.25.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-pack
ages (2.0.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dis
t-packages (1.2.2)
Collecting tensorflow_decision_forests
  Downloading tensorflow_decision_forests-1.9.0-cp310-cp310-manylinux_2_17_x
86_64.manylinux2014_x86_64.whl (15.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.5/15.5 MB 21.5 MB/s eta 0:0
0:00
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-
packages (3.7.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pyth
on3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dis
t-packages (from pandas) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/d
ist-packages (from pandas) (2024.1)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dis
t-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/di
st-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python
3.10/dist-packages (from scikit-learn) (3.4.0)
Collecting tensorflow~=2.16.1 (from tensorflow_decision_forests)
  Downloading tensorflow-2.16.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2
014_x86_64.whl (589.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 589.8/589.8 MB 1.0 MB/s eta 0:
00:00
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-package
s (from tensorflow_decision_forests) (1.16.0)
Requirement already satisfied: absl-py in /usr/local/lib/python3.10/dist-pac
kages (from tensorflow_decision_forests) (1.4.0)
Requirement already satisfied: wheel in /usr/local/lib/python3.10/dist-packa
ges (from tensorflow_decision_forests) (0.43.0)
Collecting wurlitzer (from tensorflow_decision_forests)
  Downloading wurlitzer-3.0.3-py3-none-any.whl (7.3 kB)
Collecting tf-keras~=2.16 (from tensorflow_decision_forests)
  Downloading tf_keras-2.16.0-py3-none-any.whl (1.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 27.3 MB/s eta 0:00:
00
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dis
t-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib) (4.50.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/
dist-packages (from matplotlib) (24.0)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/di
st-packages (from matplotlib) (9.4.0)
```

```
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib) (3.1.2)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.1
0/dist-packages (from tensorflow~=2.16.1->tensorflow_decision_forests) (1.6.
3)
Requirement already satisfied: flatbuffers>=23.5.26 in /usr/local/lib/python
3.10/dist-packages (from tensorflow~=2.16.1->tensorflow_decision_forests) (2
4.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/l
ocal/lib/python3.10/dist-packages (from tensorflow~=2.16.1->tensorflow_decis
ion_forests) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python
3.10/dist-packages (from tensorflow~=2.16.1->tensorflow_decision_forests)
(0.2.0)
Collecting h5py>=3.10.0 (from tensorflow~=2.16.1->tensorflow_decision_forest
s)
  Downloading h5py-3.10.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x8
6_64.whl (4.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.8/4.8 MB 51.4 MB/s eta 0:00:
00
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.1
0/dist-packages (from tensorflow~=2.16.1->tensorflow_decision_forests) (18.
1.1)
Collecting ml-dtypes~=0.3.1 (from tensorflow~=2.16.1->tensorflow_decision_fo
rests)
  Downloading ml_dtypes-0.3.2-cp310-cp310-manylinux_2_17_x86_64.manylinux201
4_x86_64.whl (2.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.2/2.2 MB 46.7 MB/s eta 0:00:
00
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.1
0/dist-packages (from tensorflow~=2.16.1->tensorflow_decision_forests) (3.3.
0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!
=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/python3.10/dist-packag
es (from tensorflow~=2.16.1->tensorflow_decision_forests) (3.20.3)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python
3.10/dist-packages (from tensorflow~=2.16.1->tensorflow_decision_forests)
(2.31.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-
packages (from tensorflow~=2.16.1->tensorflow_decision_forests) (67.7.2)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.1
0/dist-packages (from tensorflow~=2.16.1->tensorflow_decision_forests) (2.4.
0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/py
thon3.10/dist-packages (from tensorflow~=2.16.1->tensorflow_decision_forest
s) (4.10.0)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.10/di
st-packages (from tensorflow~=2.16.1->tensorflow_decision_forests) (1.14.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python
3.10/dist-packages (from tensorflow~=2.16.1->tensorflow_decision_forests)
(1.62.1)
Collecting tensorboard<2.17,>=2.16 (from tensorflow~=2.16.1->tensorflow_deci
sion_forests)
  Downloading tensorboard-2.16.2-py3-none-any.whl (5.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.5/5.5 MB 41.5 MB/s eta 0:00:
00
```

```
Collecting Keras
  Downloading keras-3.2.0-py3-none-any.whl (1.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 43.0 MB/s eta 0:00:
00
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/
local/lib/python3.10/dist-packages (from tensorflow~=2.16.1->tensorflow_deci
sion_forests) (0.36.0)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packag
es (from Keras) (13.7.1)
Collecting namex (from Keras)
  Downloading namex-0.0.7-py3-none-any.whl (5.8 kB)
Collecting optree (from Keras)
  Downloading optree-0.11.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_
x86_64.whl (311 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 311.2/311.2 kB 23.0 MB/s eta
0:00:00
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/pytho
n3.10/dist-packages (from rich->Keras) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/pyt
hon3.10/dist-packages (from rich->Keras) (2.16.1)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-
packages (from markdown-it-py>=2.2.0->rich->Keras) (0.1.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/py
thon3.10/dist-packages (from requests<3,>=2.21.0->tensorflow~=2.16.1->tensor
flow_decision_forests) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dis
t-packages (from requests<3,>=2.21.0->tensorflow~=2.16.1->tensorflow_decisio
n_forests) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.
10/dist-packages (from requests<3,>=2.21.0->tensorflow~=2.16.1->tensorflow_d
ecision_forests) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.
10/dist-packages (from requests<3,>=2.21.0->tensorflow~=2.16.1->tensorflow_d
ecision_forests) (2024.2.2)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/
dist-packages (from tensorboard<2.17,>=2.16->tensorflow~=2.16.1->tensorflow_
decision_forests) (3.6)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /us
r/local/lib/python3.10/dist-packages (from tensorboard<2.17,>=2.16->tensorfl
ow~=2.16.1->tensorflow_decision_forests) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/
dist-packages (from tensorboard<2.17,>=2.16->tensorflow~=2.16.1->tensorflow_
decision_forests) (3.0.2)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.1
0/dist-packages (from werkzeug>=1.0.1->tensorboard<2.17,>=2.16->tensorflow~=
2.16.1->tensorflow_decision_forests) (2.1.5)
Installing collected packages: namex, wurlitzer, optree, ml-dtypes, h5py, te
nsorboard, Keras, tensorflow, tf-keras, tensorflow_decision_forests
  Attempting uninstall: ml-dtypes
    Found existing installation: ml-dtypes 0.2.0
    Uninstalling ml-dtypes-0.2.0:
      Successfully uninstalled ml-dtypes-0.2.0
  Attempting uninstall: h5py
    Found existing installation: h5py 3.9.0
    Uninstalling h5py-3.9.0:
      Successfully uninstalled h5py-3.9.0
```

```
  Attempting uninstall: tensorboard
    Found existing installation: tensorboard 2.15.2
    Uninstalling tensorboard-2.15.2:
      Successfully uninstalled tensorboard-2.15.2
  Attempting uninstall: Keras
    Found existing installation: keras 2.15.0
    Uninstalling keras-2.15.0:
      Successfully uninstalled keras-2.15.0
  Attempting uninstall: tensorflow
    Found existing installation: tensorflow 2.15.0
    Uninstalling tensorflow-2.15.0:
      Successfully uninstalled tensorflow-2.15.0
  Attempting uninstall: tf-keras
    Found existing installation: tf_keras 2.15.1
    Uninstalling tf_keras-2.15.1:
      Successfully uninstalled tf_keras-2.15.1
Successfully installed Keras-3.2.0 h5py-3.10.0 ml-dtypes-0.3.2 namex-0.0.7 o
ptree-0.11.0 tensorboard-2.16.2 tensorflow-2.16.1 tensorflow_decision_forest
s-1.9.0 tf-keras-2.16.0 wurlitzer-3.0.3
```

In [3]:

```python
import matplotlib.pyplot as plt
import csv
import sqlite3
import time
import numpy as np
import pandas as pd
from pprint import pprint
from pandas import DataFrame
from sklearn.model_selection import KFold
from sklearn.decomposition import PCA
from sklearn.metrics import auc
from sklearn.metrics import roc_auc_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score
from sklearn.metrics import precision_recall_curve
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import label_binarize
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import GradientBoostingClassifier
from keras.models import Model
from keras.layers import Dense, Dropout, Input, Activation, BatchNormalizati
from keras.callbacks import EarlyStopping
import warnings
warnings.filterwarnings('ignore')
```

Data

The authors collected 3844 FDA approved drugs and 567 experimental drugs, their corresponding DDIs as well as four features of drugs: chemical substructures, targets, pathways and enzymes from DrugBank. Then StanfordNLP tool was used to construct

the events: (drugA, drugB, mechanism, action), where the mechanism means the effect of drugs in terms of the metabolism, the serum concentration, the therapeutic efficacy and so on. The action represents the effectiveness change as the result of the DDI.

Source data can be found at:
https://github.com/YifanDengWHU/DDIMDL/blob/master/event.db

In [4]:
```
event_num = 65
vector_size = 572
seed = 0
```

In [5]:
```
DATA_FILE="event.db"

conn = sqlite3.connect(DATA_FILE)
df_drug = pd.read_sql('select * from drug;', conn)
df_event = pd.read_sql('select * from event_number;', conn)
df_interaction = pd.read_sql('select * from event;', conn)
```

In [6]:
```
# Features to use
feature_list = ['smile', 'target', 'enzyme']

featureName="+".join(feature_list)
set_name = '+'.join(feature_list)
all_matrix = []
drugList=[]

extraction = pd.read_sql('select * from extraction;', conn)
mechanism = extraction['mechanism']
action = extraction['action']
drugA = extraction['drugA']
drugB = extraction['drugB']
```

In [7]:
```
df_drug
```

Out[7]:

	index	id	target	
0	0	DB01296	P14780|Q00653|P01375|P01579|P33673	
1	1	DB09230	Q02641	
2	2	DB05812	P05093	P08684|Q0
3	3	DB01195	Q14524|P35499|Q12809	
4	4	DB00201	P30542|P29274|Q07343|P21817|BE0004922|P78527|O...	P20815|P0
...	...	...	...	
567	567	DB01587	P30536|P14867|P18505|Q8N1C3|O14764|P78334	
568	568	DB00448	P20648|P10636	P33261|P11
569	569	DB00559	P25101|P24530	
570	570	DB04953	O43526|O43525|P56696|Q9NR82	P22
571	571	DB08865	Q9UM73|P08581	

572 rows × 7 columns

In [8]:
```
df_event
```

Out[8]:

	event	number
0	The metabolism of name can be decreased when c...	19620
1	The risk or severity of adverse effects can be...	18992
2	The serum concentration of name can be increas...	11292
3	The serum concentration of name can be decreas...	4772
4	The therapeutic efficacy of name can be decrea...	2624
...	...	...
60	The risk of a hypersensitivity reaction to nam...	10
61	name may increase the hyperglycemic activities...	10
62	name may increase the hypocalcemic activities ...	10
63	name may increase the myelosuppressive activit...	10
64	name may increase the vasodilatory activities ...	10

65 rows × 2 columns

In [9]:
```
df_interaction
```

Out[9]:

	index	id1	name1	id2	name2	interaction
0	0	DB12001	Abemaciclib	DB01118	Amiodarone	The risk or severity of adverse effects can be...
1	1	DB12001	Abemaciclib	DB11901	Apalutamide	The serum concentration of Abemaciclib can be ...
2	2	DB12001	Abemaciclib	DB00673	Aprepitant	The serum concentration of Abemaciclib can be ...
3	3	DB12001	Abemaciclib	DB00289	Atomoxetine	The metabolism of Abemaciclib can be decreased...
4	4	DB12001	Abemaciclib	DB00188	Bortezomib	The metabolism of Abemaciclib can be decreased...
...	...	...	...	...	...	...
37259	37259	DB01149	Nefazodone	DB09048	Netupitant	The serum concentration of Nefazodone can be i...
37260	37260	DB01149	Nefazodone	DB00622	Nicardipine	The metabolism of Nefazodone can be decreased ...
37261	37261	DB11828	Neratinib	DB09048	Netupitant	The serum concentration of Neratinib can be in...
37262	37262	DB09048	Netupitant	DB00622	Nicardipine	The serum concentration of Netupitant can be i...
37263	37263	DB00622	Nicardipine	DB00788	Naproxen	The metabolism of Nicardipine can be decreased...

37264 rows × 6 columns

In [10]:
```
extraction
```

Out[10]:

	index	mechanism	action	drugA	drugB
0	0	The risk or severity of adverse effects	increase	Abemaciclib	Amiodarone
1	1	The serum concentration	decrease	Abemaciclib	Apalutamide
2	2	The serum concentration	increase	Abemaciclib	Aprepitant
3	3	The metabolism	decrease	Abemaciclib	Atomoxetine
4	4	The metabolism	decrease	Abemaciclib	Bortezomib
...	...	...	...	...	...
37259	37259	The serum concentration	increase	Nefazodone	Netupitant
37260	37260	The metabolism	decrease	Nefazodone	Nicardipine
37261	37261	The serum concentration	increase	Neratinib	Netupitant
37262	37262	The serum concentration	increase	Netupitant	Nicardipine
37263	37263	The metabolism	decrease	Nicardipine	Naproxen

37264 rows × 5 columns

In [11]:

```
def prepare(df_drug, feature_list, vector_size,mechanism,action,drugA,drugB)
    d_label = {}
    d_feature = {}
    # Transfrom the interaction event to number
    # Splice the features
    d_event=[]
    for i in range(len(mechanism)):
        d_event.append(mechanism[i]+" "+action[i])
    label_value = 0
    count={}
    for i in d_event:
        if i in count:
            count[i]+=1
        else:
            count[i]=1
    list1 = sorted(count.items(), key=lambda x: x[1],reverse=True)
    for i in range(len(list1)):
        d_label[list1[i][0]]=i
    vector = np.zeros((len(np.array(df_drug['name']).tolist()), 0), dtype=fl
    for i in feature_list:
        vector = np.hstack((vector, feature_vector(i, df_drug, vector_size))
    # Transfrom the drug ID to feature vector
    for i in range(len(np.array(df_drug['name']).tolist())):
        d_feature[np.array(df_drug['name']).tolist()[i]] = vector[i]
    # Use the dictionary to obtain feature vector and label
    new_feature = []
    new_label = []
    name_to_id = {}
    for i in range(len(d_event)):
        new_feature.append(np.hstack((d_feature[drugA[i]], d_feature[drugB[i
```

```
        new_label.append(d_label[d_event[i]])
    new_feature = np.array(new_feature)
    new_label = np.array(new_label)
    return (new_feature, new_label, event_num)

def feature_vector(feature_name, df, vector_size):
    # df are the 572 kinds of drugs
    # Jaccard Similarity
    def Jaccard(matrix):
        matrix = np.mat(matrix)
        numerator = matrix * matrix.T
        denominator = np.ones(np.shape(matrix)) * matrix.T + matrix * np.one
        return numerator / denominator

    all_feature = []
    drug_list = np.array(df[feature_name]).tolist()
    # Features for each drug, for example, when feature_name is target, drug
    for i in drug_list:
        for each_feature in i.split('|'):
            if each_feature not in all_feature:
                all_feature.append(each_feature)  # obtain all the features
    feature_matrix = np.zeros((len(drug_list), len(all_feature)), dtype=floa
    df_feature = DataFrame(feature_matrix, columns=all_feature)  # Consrtuct
    for i in range(len(drug_list)):
        for each_feature in df[feature_name].iloc[i].split('|'):
            df_feature[each_feature].iloc[i] = 1
    sim_matrix = Jaccard(np.array(df_feature))

    sim_matrix1 = np.array(sim_matrix)
    count = 0
    pca = PCA(n_components=vector_size)  # PCA dimension
    pca.fit(np.asarray(sim_matrix))
    sim_matrix = pca.transform(np.asarray(sim_matrix))
    return sim_matrix
```

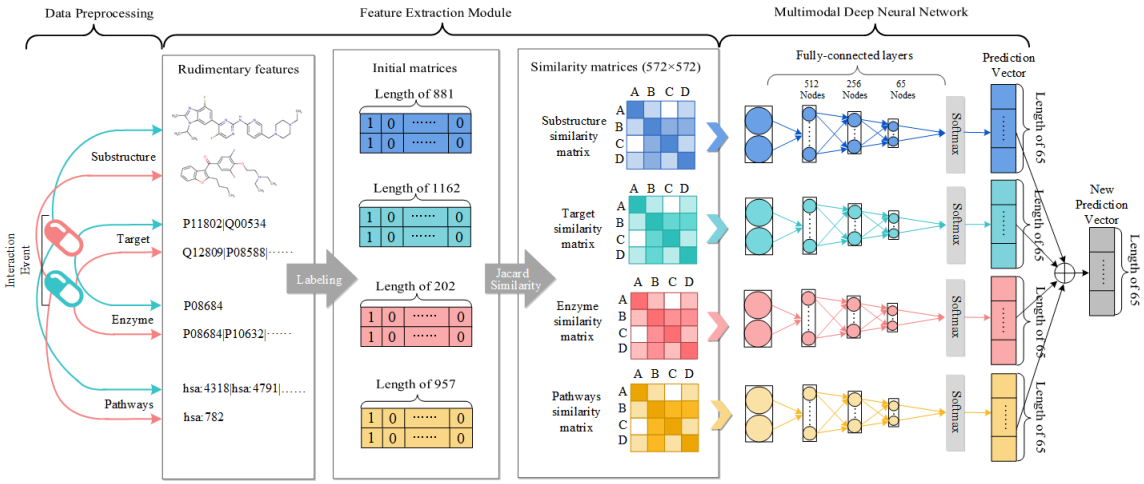
In [12]:

```
for feature in feature_list:
    pprint(f'feature feat={feature}')
    new_feature, new_label, event_num = prepare(df_drug, [feature], vector_s
    all_matrix.append(new_feature)
all_matrix
```

```
'feature feat=smile'
'feature feat=target'
'feature feat=enzyme'
```

Out[12]:
```
[array([[-5.50822011e-01,  3.16967827e+00, -1.37253742e-01, ...,
         -5.55111512e-17, -1.38777878e-17, -7.28583860e-17],
        [-5.50822011e-01,  3.16967827e+00, -1.37253742e-01, ...,
         -4.85722573e-17, -4.16333634e-17,  1.04083409e-16],
        [-5.50822011e-01,  3.16967827e+00, -1.37253742e-01, ...,
         -6.93889390e-18, -4.59701721e-17, -4.16333634e-17],
        ...,
        [-1.09680344e+00,  1.38541285e+00, -1.38446925e+00, ...,
          2.77555756e-17, -2.60208521e-17,  1.82498331e-16],
        [-1.35282107e+00,  2.30799232e+00,  1.51359574e-01, ...,
         -4.85722573e-17,  2.42861287e-17, -6.93889390e-17],
        [-2.07144651e+00, -1.71437659e-01,  3.37111947e-01, ...,
         -5.55111512e-17,  0.00000000e+00, -1.23165367e-16]]),
 array([[-5.41580850e-02, -1.26844824e-01, -1.30110552e-01, ...,
          3.12250226e-17,  6.93889390e-17, -3.46944695e-18],
        [-5.41580850e-02, -1.26844824e-01, -1.30110552e-01, ...,
         -2.08166817e-17,  2.08166817e-17, -6.93889390e-17],
        [-5.41580850e-02, -1.26844824e-01, -1.30110552e-01, ...,
          1.14491749e-16, -1.56125113e-17, -2.08166817e-17],
        ...,
        [-5.87168480e-02, -1.41102388e-01, -1.44321451e-01, ...,
          1.14491749e-16, -1.56125113e-17, -2.08166817e-17],
        [-5.54142334e-02, -1.28724805e-01, -1.33400624e-01, ...,
          7.63278329e-17,  1.17961196e-16, -1.86482774e-17],
        [-2.47428700e-01,  2.35510805e-01, -1.41487960e-01, ...,
          1.38777878e-16,  1.45716772e-16, -1.38777878e-16]]),
 array([[ 6.20485105e+00, -2.07809091e+00,  5.54920630e-01, ...,
         -5.55111512e-17,  3.98986399e-17, -1.14491749e-16],
        [ 6.20485105e+00, -2.07809091e+00,  5.54920630e-01, ...,
         -2.49800181e-16,  6.82613688e-16,  6.10622664e-16],
        [ 6.20485105e+00, -2.07809091e+00,  5.54920630e-01, ...,
          0.00000000e+00, -1.99493200e-16, -2.77555756e-17],
        ...,
        [ 2.32258134e+00, -1.50270110e+00,  1.61843382e-01, ...,
         -3.12250226e-16, -6.50521303e-17,  2.08166817e-17],
        [ 9.60115190e-01,  1.89851311e+00,  4.31840542e-01, ...,
          4.16333634e-16,  2.17707796e-16,  4.57099636e-16],
        [ 1.31606213e-01,  3.13094689e+00, -1.86506774e+00, ...,
         -7.09068221e-16, -5.84757665e-17, -2.91433544e-16]])]
```

Model



Layers	Configuration	Activation Function	Output Dimension (batch, feature)
Fully connected	Input size: 1144, Output size: 512	ReLU	(128, 512)
Batch Normalization	-	-	(128, 512)
Dropout	Dropout rate: 0.3	-	(128, 512)
Fully connected	Input size: 512, Output size: 256	ReLU	(128, 256)
Batch Normalization	-	-	(128, 256)
Dropout	Dropout rate: 0.3	-	(128, 256)
Fully connected	Input size: 256, Output size: 65	-	(128, 65)
Activation	Softmax	-	(128, 65)

In [13]:
```
droprate = 0.3

def DNN():
    train_input = Input(shape=(vector_size * 2,), name='Inputlayer')
    train_in = Dense(512, activation='relu')(train_input)
    train_in = BatchNormalization()(train_in)
    train_in = Dropout(droprate)(train_in)
    train_in = Dense(256, activation='relu')(train_in)
    train_in = BatchNormalization()(train_in)
    train_in = Dropout(droprate)(train_in)
    train_in = Dense(event_num)(train_in)
    out = Activation('softmax')(train_in)
    model = Model(train_input, out)
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics

    return model
```

Results

Computational requirements

The project notebook was written in a Colab notebook with CPU only environment. In additon, the local notebook was run on a 2.6 GHz 6-Core Intel Core i7 Macbook Pro. The main DNN model was relatively quick, averaging around 3mins for 3-fold, 3-feature, 3 epochs setting. No GPU was used.

Training (DNN)

In this section, we use train the proposed DNN model from the original paper, and measure the performance for each fold, feature, and epoch. The measurement is plotted into k-fold number of rows, each row represents the corresponding performance for the fold. Each column contains the model performance trained with the corresponding feature. And finally, within each plot, the accuracy and losses over epochs are drawn.

In [14]:
```
epochs = 3
CV = 3 # Kfold
```

In [15]:
```
import tensorflow_decision_forests as tfdf
def get_index(label_matrix, event_num, seed, CV):
    index_all_class = np.zeros(len(label_matrix))
    for j in range(event_num):
        index = np.where(label_matrix == j)
        kf = KFold(n_splits=CV, shuffle=True, random_state=seed)
        k_num = 0
        for train_index, test_index in kf.split(range(len(index[0]))):
            index_all_class[index[0][test_index]] = k_num
            k_num += 1

    return index_all_class

def multiclass_precision_recall_curve(y_true, y_score):
    y_true = y_true.ravel()
    y_score = y_score.ravel()
    if y_true.ndim == 1:
        y_true = y_true.reshape((-1, 1))
    if y_score.ndim == 1:
        y_score = y_score.reshape((-1, 1))
    y_true_c = y_true.take([0], axis=1).ravel()
    y_score_c = y_score.take([0], axis=1).ravel()
    precision, recall, pr_thresholds = precision_recall_curve(y_true_c, y_sc
    return (precision, recall, pr_thresholds)

def roc_aupr_score(y_true, y_score, average="macro"):
    def _binary_roc_aupr_score(y_true, y_score):
```

```
        precision, recall, pr_thresholds = precision_recall_curve(y_true, y_
        return auc(recall, precision)

    def _average_binary_score(binary_metric, y_true, y_score, average):  # y
        if average == "binary":
            return binary_metric(y_true, y_score)
        if average == "micro":
            y_true = y_true.ravel()
            y_score = y_score.ravel()
        if y_true.ndim == 1:
            y_true = y_true.reshape((-1, 1))
        if y_score.ndim == 1:
            y_score = y_score.reshape((-1, 1))
        n_classes = y_score.shape[1]
        score = np.zeros((n_classes,))
        for c in range(n_classes):
            y_true_c = y_true.take([c], axis=1).ravel()
            y_score_c = y_score.take([c], axis=1).ravel()
            score[c] = binary_metric(y_true_c, y_score_c)
        return np.average(score)

    return _average_binary_score(_binary_roc_aupr_score, y_true, y_score, av


def evaluate(pred_type, pred_score, y_test, event_num, set_name):
    # 11 metric types for overall metrics
    all_eval_type = 11
    # 6 metric types for per-event based metrics
    each_eval_type = 6

    result_all = DataFrame(columns=['Accuracy', 'AUPR_micro', 'AUPR_macro',
                                    'F1_macro', 'Precision_micro', 'Precisic
    result_eve = DataFrame(columns=['Event#', 'Accuracy', 'AUPR', 'AUROC', '

    y_one_hot = label_binarize(y_test, classes=np.arange(event_num))
    pred_one_hot = label_binarize(pred_type, classes=np.arange(event_num))
    precision, recall, th = multiclass_precision_recall_curve(y_one_hot, pre

    result_all.loc[0, 'Accuracy'] = accuracy_score(y_test, pred_type)
    result_all.loc[0, 'AUPR_micro'] = roc_aupr_score(y_one_hot, pred_score,
    result_all.loc[0, 'AUPR_macro'] = roc_aupr_score(y_one_hot, pred_score,
    result_all.loc[0, 'AUROC_micro'] = roc_auc_score(y_one_hot, pred_score,
    result_all.loc[0, 'AUROC_macro'] = roc_auc_score(y_one_hot, pred_score,
    result_all.loc[0, 'F1_micro'] = f1_score(y_test, pred_type, average='mic
    result_all.loc[0, 'F1_macro'] = f1_score(y_test, pred_type, average='mac
    result_all.loc[0, 'Precision_micro'] = precision_score(y_test, pred_type
    result_all.loc[0, 'Precision_macro'] = precision_score(y_test, pred_type
    result_all.loc[0, 'Recall_micro'] = recall_score(y_test, pred_type, aver
    result_all.loc[0, 'Recall_macro'] = recall_score(y_test, pred_type, aver

    for i in range(event_num):
        result_eve.loc[i, 'Event#'] = i
        result_eve.loc[i, 'Accuracy'] = accuracy_score(y_one_hot.take([i], a
                                                       pred_one_hot.take([i]
        result_eve.loc[i, 'AUPR'] = roc_aupr_score(y_one_hot.take([i], axis=
                                                   pred_one_hot.take([i], ax
```

```
        result_eve.loc[i, 'AUROC'] = roc_auc_score(y_one_hot.take([i], axis=
                                                   pred_one_hot.take([i], axi
        result_eve.loc[i, 'F1'] = f1_score(y_one_hot.take([i], axis=1).ravel
                                           average='binary')
        result_eve.loc[i, 'Precision'] = precision_score(y_one_hot.take([i],
                                                         pred_one_hot.take([i
        result_eve.loc[i, 'Recall'] = recall_score(y_one_hot.take([i], axis=
                                                   pred_one_hot.take([i], axi
    return [result_all, result_eve]

def cross_validation(feature_matrix, label_matrix, clf_type, event_num, seed
    all_eval_type = 11
    result_all = np.zeros((all_eval_type, 1), dtype=float)
    each_eval_type = 6
    result_eve = np.zeros((event_num, each_eval_type), dtype=float)
    y_true = np.array([])
    y_pred = np.array([])
    y_score = np.zeros((0, event_num), dtype=float)
    index_all_class = get_index(label_matrix, event_num, seed, CV)
    matrix = []
    if type(feature_matrix) != list:
        matrix.append(feature_matrix)
        feature_matrix = matrix
    if clf_type == 'DDIMDL':
      # plot settings
      xlabels = range(1, epochs + 1)
      fig, ax = plt.subplots(CV, len(feature_matrix))
    for k in range(CV):
        pprint(f'k-fold k={k+1}')
        train_index = np.where(index_all_class != k)
        test_index = np.where(index_all_class == k)
        pred = np.zeros((len(test_index[0]), event_num), dtype=float)
        for i in range(len(feature_matrix)):
            pprint(f'feature feat={feature_list[i]}')
            x_train = feature_matrix[i][train_index]
            x_test = feature_matrix[i][test_index]
            y_train = label_matrix[train_index]
            # one-hot encoding
            y_train_one_hot = np.array(y_train)
            y_train_one_hot = (np.arange(y_train_one_hot.max() + 1) == y_tra
            y_test = label_matrix[test_index]
            # one-hot encoding
            y_test_one_hot = np.array(y_test)
            y_test_one_hot = (np.arange(y_test_one_hot.max() + 1) == y_test[
            if clf_type == 'DDIMDL':
                dnn = DNN()
                early_stopping = EarlyStopping(monitor='val_loss', patience=
                history = dnn.fit(x_train, y_train_one_hot, batch_size=128,
                        callbacks=[early_stopping])
                pred += dnn.predict(x_test)
                # plot accuracy and loss over epochs
                acc = history.history['accuracy']
                loss = history.history['loss']
                ax[k, i].plot(xlabels, acc, label=f'Accuracy')
                ax[k, i].plot(xlabels, loss, label=f'Loss')
                ax[k, i].title.set_text(f'k={k} feat={feature_list[i]}')
```

```
                ax[k, i].legend()
                continue
            elif clf_type == 'RF':
                # all decision forests algorithms train with only 1 epoch
                clf = RandomForestClassifier(n_estimators=100)
            elif clf_type == 'GBDT':
                clf = GradientBoostingClassifier()
            elif clf_type == 'SVM':
                clf = SVC(probability=True)
            elif clf_type == 'KNN':
                clf = KNeighborsClassifier(n_neighbors=4)
            else:
                clf = LogisticRegression()
            clf.fit(x_train, y_train)
            pred += clf.predict_proba(x_test)
        pred_score = pred / len(feature_matrix)
        pred_type = np.argmax(pred_score, axis=1)
        y_true = np.hstack((y_true, y_test))
        y_pred = np.hstack((y_pred, pred_type))
        y_score = np.row_stack((y_score, pred_score))

    if clf_type == 'DDIMDL':
      # plot the graphs
      fig.supxlabel(f'Epochs ({epochs})')
      fig.supylabel('Accuracy/Loss')
      fig.suptitle('Training accuracy and loss', fontsize=16)
      plt.show()

    result_all, result_eve = evaluate(y_pred, y_score, y_true, event_num, se

    return result_all, result_eve
```

In [16]:

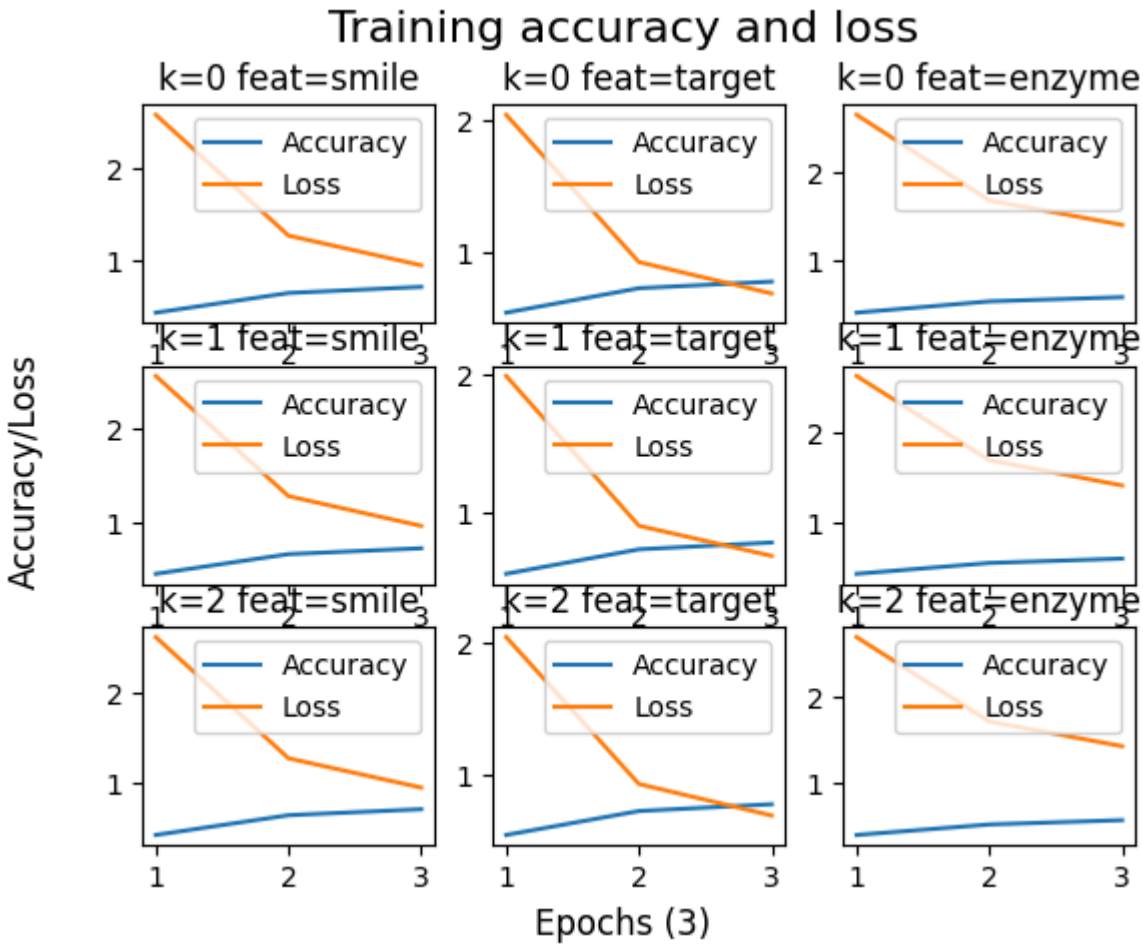
```
result_all = {}
result_eve = {}
```

In [17]:

```
# Classifier to use: DDIMDL, RF, KNN, LR
DDIMDL = 'DDIMDL'
pprint(f'clf model={DDIMDL}')
all_result, each_result = cross_validation(all_matrix, new_label, DDIMDL, ev
                                           set_name)
result_all[DDIMDL] = all_result
result_eve[DDIMDL] = each_result
```

```
'clf model=DDIMDL'
'k-fold k=1'
'feature feat=smile'
Epoch 1/3
194/194 ━━━━━━━━━━━━━━━━━━━━ 11s 34ms/step - accuracy: 0.3010 - loss: 3.4289
- val_accuracy: 0.4041 - val_loss: 2.0551
Epoch 2/3
194/194 ━━━━━━━━━━━━━━━━━━━━ 5s 28ms/step - accuracy: 0.6405 - loss: 1.3498
- val_accuracy: 0.6013 - val_loss: 1.3590
Epoch 3/3
194/194 ━━━━━━━━━━━━━━━━━━━━ 6s 29ms/step - accuracy: 0.7171 - loss: 0.9578
- val_accuracy: 0.7172 - val_loss: 0.8954
389/389 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step
'feature feat=target'
Epoch 1/3
194/194 ━━━━━━━━━━━━━━━━━━━━ 9s 36ms/step - accuracy: 0.3992 - loss: 2.9130
- val_accuracy: 0.5197 - val_loss: 1.9127
Epoch 2/3
194/194 ━━━━━━━━━━━━━━━━━━━━ 9s 28ms/step - accuracy: 0.7246 - loss: 0.9751
- val_accuracy: 0.6414 - val_loss: 1.1518
Epoch 3/3
194/194 ━━━━━━━━━━━━━━━━━━━━ 10s 28ms/step - accuracy: 0.7839 - loss: 0.6965
- val_accuracy: 0.7667 - val_loss: 0.7060
389/389 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step
'feature feat=enzyme'
Epoch 1/3
194/194 ━━━━━━━━━━━━━━━━━━━━ 7s 28ms/step - accuracy: 0.3271 - loss: 3.2750
- val_accuracy: 0.4818 - val_loss: 2.1082
Epoch 2/3
194/194 ━━━━━━━━━━━━━━━━━━━━ 7s 34ms/step - accuracy: 0.5256 - loss: 1.7618
- val_accuracy: 0.5547 - val_loss: 1.5394
Epoch 3/3
194/194 ━━━━━━━━━━━━━━━━━━━━ 10s 31ms/step - accuracy: 0.5800 - loss: 1.4253
- val_accuracy: 0.6103 - val_loss: 1.2984
389/389 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step
'k-fold k=2'
'feature feat=smile'
Epoch 1/3
195/195 ━━━━━━━━━━━━━━━━━━━━ 11s 41ms/step - accuracy: 0.3022 - loss: 3.3972
- val_accuracy: 0.3536 - val_loss: 2.2282
Epoch 2/3
195/195 ━━━━━━━━━━━━━━━━━━━━ 6s 29ms/step - accuracy: 0.6360 - loss: 1.3700
- val_accuracy: 0.5759 - val_loss: 1.3864
Epoch 3/3
195/195 ━━━━━━━━━━━━━━━━━━━━ 6s 30ms/step - accuracy: 0.7139 - loss: 0.9667
- val_accuracy: 0.7276 - val_loss: 0.9084
389/389 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step
'feature feat=target'
Epoch 1/3
195/195 ━━━━━━━━━━━━━━━━━━━━ 8s 30ms/step - accuracy: 0.4126 - loss: 2.8385
- val_accuracy: 0.4957 - val_loss: 1.8865
Epoch 2/3
195/195 ━━━━━━━━━━━━━━━━━━━━ 10s 28ms/step - accuracy: 0.7287 - loss: 0.9439
- val_accuracy: 0.6300 - val_loss: 1.1627
Epoch 3/3
195/195 ━━━━━━━━━━━━━━━━━━━━ 7s 37ms/step - accuracy: 0.7865 - loss: 0.6853
```

```
- val_accuracy: 0.7593 - val_loss: 0.7446
389/389 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step
'feature feat=enzyme'
Epoch 1/3
195/195 ━━━━━━━━━━━━━━━━━━━━ 9s 36ms/step - accuracy: 0.3370 - loss: 3.2524
- val_accuracy: 0.3890 - val_loss: 2.1071
Epoch 2/3
195/195 ━━━━━━━━━━━━━━━━━━━━ 9s 28ms/step - accuracy: 0.5314 - loss: 1.7675
- val_accuracy: 0.5353 - val_loss: 1.5921
Epoch 3/3
195/195 ━━━━━━━━━━━━━━━━━━━━ 6s 33ms/step - accuracy: 0.5778 - loss: 1.4385
- val_accuracy: 0.6042 - val_loss: 1.3101
389/389 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step
'k-fold k=3'
'feature feat=smile'
Epoch 1/3
195/195 ━━━━━━━━━━━━━━━━━━━━ 9s 33ms/step - accuracy: 0.2986 - loss: 3.4315
- val_accuracy: 0.4519 - val_loss: 2.1336
Epoch 2/3
195/195 ━━━━━━━━━━━━━━━━━━━━ 10s 30ms/step - accuracy: 0.6325 - loss: 1.3865
- val_accuracy: 0.6348 - val_loss: 1.3290
Epoch 3/3
195/195 ━━━━━━━━━━━━━━━━━━━━ 5s 28ms/step - accuracy: 0.7144 - loss: 0.9701
- val_accuracy: 0.7240 - val_loss: 0.8830
388/388 ━━━━━━━━━━━━━━━━━━━━ 3s 6ms/step
'feature feat=target'
Epoch 1/3
195/195 ━━━━━━━━━━━━━━━━━━━━ 9s 35ms/step - accuracy: 0.4011 - loss: 2.8884
- val_accuracy: 0.5335 - val_loss: 2.0317
Epoch 2/3
195/195 ━━━━━━━━━━━━━━━━━━━━ 6s 31ms/step - accuracy: 0.7246 - loss: 0.9704
- val_accuracy: 0.6720 - val_loss: 1.1055
Epoch 3/3
195/195 ━━━━━━━━━━━━━━━━━━━━ 10s 31ms/step - accuracy: 0.7822 - loss: 0.7033
- val_accuracy: 0.7752 - val_loss: 0.6886
388/388 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step
'feature feat=enzyme'
Epoch 1/3
195/195 ━━━━━━━━━━━━━━━━━━━━ 9s 33ms/step - accuracy: 0.3296 - loss: 3.3035
- val_accuracy: 0.4812 - val_loss: 1.9900
Epoch 2/3
195/195 ━━━━━━━━━━━━━━━━━━━━ 5s 25ms/step - accuracy: 0.5158 - loss: 1.8066
- val_accuracy: 0.5587 - val_loss: 1.5068
Epoch 3/3
195/195 ━━━━━━━━━━━━━━━━━━━━ 6s 31ms/step - accuracy: 0.5762 - loss: 1.4459
- val_accuracy: 0.6120 - val_loss: 1.2746
388/388 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step
```



In [18]:
```
result_all['DDIMDL']
```

Out[18]:

	Accuracy	AUPR_micro	AUPR_macro	AUROC_micro	AUROC_macro	F1_micro	F1_
0	0.771925	0.839645	0.63618	0.994526	0.977938	0.771925	0.3

In [19]:
```
result_eve['DDIMDL']
```

Out[19]:

	Event#	Accuracy	AUPR	AUROC	F1	Precision	Recall
0	0	0.885412	0.835419	0.903235	0.812143	0.714396	0.940877
1	1	0.901165	0.84131	0.900213	0.822446	0.758424	0.898273
2	2	0.933985	0.792934	0.85009	0.770093	0.815196	0.72972
3	3	0.979498	0.840099	0.886167	0.82954	0.886927	0.779128
4	4	0.98363	0.755926	0.816	0.732221	0.863354	0.635671
...	...	...	...	...	...	...	...
60	60	0.999866	0.500067	0.5	0.0	0.0	0.0
61	61	0.999893	0.600054	0.6	0.333333	1.0	0.2
62	62	0.999866	0.500067	0.5	0.0	0.0	0.0
63	63	0.999866	0.500067	0.5	0.0	0.0	0.0
64	64	0.999866	0.500067	0.5	0.0	0.0	0.0

65 rows × 7 columns

Ablation Analysis: Training (RF)

DO NOT RUN FOR GRADING

In [20]:

```
RF = 'RF'
pprint(f'clf model={RF}')
all_result, each_result = cross_validation(all_matrix, new_label, RF, event_
                                           set_name)
result_all[RF] = all_result
result_eve[RF] = each_result
```

```
'clf model=RF'
'k-fold k=1'
'feature feat=smile'
'feature feat=target'
'feature feat=enzyme'
'k-fold k=2'
'feature feat=smile'
'feature feat=target'
'feature feat=enzyme'
'k-fold k=3'
'feature feat=smile'
'feature feat=target'
'feature feat=enzyme'
```

In [21]:

```
result_all['RF']
```

Out[21]:

	Accuracy	AUPR_micro	AUPR_macro	AUROC_micro	AUROC_macro	F1_micro	F1_
0	0.761137	0.834016	0.634219	0.994702	0.971648	0.761137	0.4

In [22]:
```
result_eve['RF']
```

Out[22]:

	Event#	Accuracy	AUPR	AUROC	F1	Precision	Recall
0	0	0.890511	0.834925	0.896639	0.81392	0.736464	0.909582
1	1	0.891423	0.828325	0.891598	0.807205	0.737163	0.891955
2	2	0.931301	0.784758	0.847345	0.762259	0.80125	0.726886
3	3	0.976251	0.813489	0.869986	0.801347	0.862736	0.748114
4	4	0.980034	0.696522	0.759425	0.648061	0.854115	0.522104
...	...	...	...	...	...	...	...
60	60	0.999946	0.800027	0.8	0.75	1.0	0.6
61	61	1.0	1.0	1.0	1.0	1.0	1.0
62	62	0.999893	0.600054	0.6	0.333333	1.0	0.2
63	63	0.999866	0.500067	0.5	0.0	0.0	0.0
64	64	0.999866	0.500067	0.5	0.0	0.0	0.0

65 rows × 7 columns

Ablation Analysis: Training (GBDT)

DO NOT RUN FOR GRADING

In [ ]:
```
GBDT = 'GBDT'
pprint(f'clf model={GBDT}')
all_result, each_result = cross_validation(all_matrix, new_label, GBDT, ever
                                           set_name)
result_all[GBDT] = all_result
result_eve[GBDT] = each_result
```

In [ ]:
```
result_all['GBDT']
```

In [ ]:
```
result_eve['GBDT']
```

Ablation Analysis: Training (SVM)

DO NOT RUN FOR GRADING

In [ ]:
```
SVM = 'SVM'
pprint(f'clf model={SVM}')
all_result, each_result = cross_validation(all_matrix, new_label, SVM, event
                                           set_name)
result_all[SVM] = all_result
result_eve[SVM] = each_result
```

In [ ]:
```
result_all['SVM']
```

In [ ]:
```
result_eve['SVM']
```

Ablation Analysis: Training (KNN)

DO NOT RUN FOR GRADING

In [ ]:
```
KNN = 'KNN'
pprint(f'clf model={KNN}')
all_result, each_result = cross_validation(all_matrix, new_label, KNN, event
                                           set_name)
result_all[KNN] = all_result
result_eve[KNN] = each_result
```

In [ ]:
```
result_all['KNN']
```

In [ ]:
```
result_eve['KNN']
```

Ablation Analysis: Training (LR)

DO NOT RUN FOR GRADING

In [ ]:
```
LR = 'LR'
pprint(f'clf model={LR}')
all_result, each_result = cross_validation(all_matrix, new_label, LR, event_
                                           set_name)
result_all[LR] = all_result
result_eve[LR] = each_result
```

In [ ]:
```
result_all['LR']
```

In [ ]:
```
result_eve['LR']
```

Model Comparison

From the previous result, it shows that with relatively low number of epochs=3, feature_num=3 and kfold=3, we still observe signifant increase of accuracy between DDIMDL model and Random Forest model, accuracy of 0.77 vs. 0.73, 4% difference. I expect the accuracy different will be larger with bigger epoch iterations in DNN.

Discussion

- In this project, I reproduced the drug-drug interaction events based on the proposed DNN(DDIMDL) model in the paper, with the pre-scraped event dataset from DrugBank.
- What is easy? Since the authors provided the collected data from DrugBank, I did not need to scrape on the DrugBank raw data.
- What is hard? Since the authors used deprecated versions of sklearn, it took me some time to set up the environment correctly with the right python version 3.7.10. But I was able to migrate their code to the latest sklearn as well as making improvement with better dependencies, including the Keras variants of the models, instead of the sklearn models.
- Suggestions: It would be great if the authors could clean up the code and make a finalized jupyter notebook report in ipynb format, clarify on the environment setups. In addition, some components are missing on GitHub from the original paper, including the DeepDDI model implementation, which was supposed to be compared with the proposed DDIMDL model.
- What will you do in next phase: For the final version of the project, I will continue exploring and do more ablation analysis to compare other classifiers with the DDIMDL model. After the project, I would like to going further to explore drug-drug interactions for more than two drugs.

References

1. Yifan Deng, Xinran Xu, Yang Qiu, Jingbo Xia, Wen Zhang, Shichao Liu, A multimodal deep learning framework for predicting drug–drug interaction events, Bioinformatics, Volume 36, Issue 15, August 2020, Pages 4316–4322, https://doi.org/10.1093/bioinformatics/btaa501