Deep Learning on Drug Discovery

Artificial Intelligence 2023

Rheina Trudy Williams - 2021380101

Cecillia Alexandra Badaruddin - 2021380098

Laura Yuan Stoma Baldo - 2021380113

Mingso Sherma Limbu - 2021380116

Submission date: 26 / 12 / 2023

Table of Content

Individual Contribution

1. Rheina Trudy Williams (2021380101) made significant contributions to the report by leading the investigation researches (analysis, design, and experiments) section, and writing it. She also designed the cover of the report and organized the final version of the report reference, while proofreading the entire report.

2. Cecillia Alexandra Badaruddin (2021380098) led the writing of the conclusions and suggestions sections, co-conducted the Investigations (Experiment and Analysis), and co-wrote the State of Art. Additionally, she meticulously formatted the reference section into scientific format.

3. Laura Yuan Stoma Baldo (2021380113) produced the Introduction portion of this report. Furthermore, she co-wrote the State of Art and the Ideas section. She organized and formatted the reference section as well.

4. Mingso Sherma Limbu (2021380116) was responsible for developing the Contents and crafting the Contributions and the Abstract. She also gathered inspirations and co-wrote the Ideas section.

Further contributions shall be split and made before our formal presentation. We held daily discussions together to enhance the overall success of this project. Therefore, the entire team collaborated and supported each other to ensure this project's completion.

Abstract

The field of drug discovery traditionally required a considerable amount of time, wealth, and manpower investment. However, it is now more efficient thanks to the advancements in computational technologies. Integration of artificial intelligence (AI) can further advance the process of drug discovery and development. By leveraging deep learning algorithms in particular; tasks such as molecular design, sustainable synthesis routes, protein structure prediction, and target identification are enabled. These models excel in uncovering hidden patterns and identifying crucial variables for drug activity. DeepChem is an open-source, python library, designed for deep learning on molecular and quantum datasets. It is deployed in various domains, most notably biology and drug development, wherein resolves pressing issues. We presented a comprehensive analysis of AI and DeepChem, addressing its capabilities and limitations. Furthermore, we delved into DeepChem's various applications that range from simulating protein-ligand interactions to lipophilicity measurement with existing examples.

In this report, some successful DeepChem-based models and their training dynamics were highlighted. Our first experiment focuses on modeling protein-ligand interactions by utilizing molecular docking, a pivotal computational tool in drug development. The second experiment employs diverse techniques to featurize protein sequences, to predict protein melting temperatures. The implications of deep learning in drug discovery are endless, offering unprecedented opportunities to develop novel therapeutics, reduce R&D costs, and address unmet medical needs. This project aims to advance our understanding of DeepChem as a pivotal tool in the realm of drug discovery, paving the way for transformative advancements in pharmaceutical innovations.

Keywords: Artificial intelligence, biotechnology, deepchem, deep learning, drug discovery

I. Introduction

Artificial intelligence (AI) has burst into the field of computer-assisted drug discovery, marking a breakthrough thanks to deep learning algorithms. These artificial neural networks, with their multiple processing layers, stand out for their ability to model complex nonlinear relationships where they can analyze input data, such as chemical properties, and predict outputs, such as biological activity, with greater accuracy than traditional methods; Recognize hidden patterns by extracting relevant information from seemingly chaotic data, identifying connections and trends that go unnoticed by the human eye or classical algorithms; Extract key features from large amounts of data identifying the most important variables that determine the desired activity of a drug [22].

These deep learning models have demonstrated performance that matches or even surpasses traditional machine learning and quantitative structure-activity relationship (QSAR) methods in drug discovery [13]. In addition, they have greatly expanded the possibilities of computer assistance in this area, opening doors to new applications such as molecular design, creating molecules with specific pharmacological properties from computational models; and Chemical synthesis planning by designing more efficient and sustainable synthesis routes for new drugs; Protein structure prediction by understanding the three-dimensional conformation of therapeutic targets to design drugs that perfectly match them and identification of macromolecular targets by discovering new molecules that can act as targets for the development of drugs against diseases that are still untreated [21, 28].

What are the challenges or difficulties that it may encounter?

Artificial intelligence (AI) promises to accelerate drug discovery, but it is not a smooth road. Collecting diverse, high-quality data sets, including data from underrepresented populations, is crucial to building reliable and unbiased models. Drug discovery often deals with rare diseases and complex biological processes, resulting in limited data sets. This can cause models to overfit the training data and fail to generalize to new molecules or targets. Also biases in data sets, such as underrepresentation of certain patient populations, can lead to unfair results, potentially missing effective drugs for marginalized groups. Also, can reinforce existing social inequalities, potentially delaying or even denying access to life-saving drugs for certain populations. It could

lead to job losses in the pharmaceutical industry, requiring careful consideration and mitigation strategies [11,26].

II. State of the art

The state of the art of deep learning in drug discovery is exciting and rapidly evolving, with advances offering immense potential to accelerate the development of new drugs and therapies.

1.Virtual screening:

- Improved accuracy and efficiency: Deep learning models can screen massive chemical libraries, identifying promising drug candidates more accurately and much faster than traditional methods. Techniques such as transfer learning, which leverages knowledge from related tasks, and active learning, which intelligently selects data points for further learning, optimizes performance and improves over time [24].
- Multi-target and multi-objective screening: Models can simultaneously consider multiple disease targets and drug properties, allowing the design of broad-spectrum drugs with optimal safety profiles [20].
- De novo drug design: Generative models are being used to create entirely new molecules with desired properties by learning from existing drugs and protein-ligand interactions. This opens new therapeutic possibilities for diseases that were previously untreatable [24].

2. Molecular modeling:

- Protein-ligand docking: Deep learning accurately predicts how molecules bind to target proteins, guiding the design of drugs with higher affinity and specificity. This improves the effectiveness of the drugs and reduces off-target side effects [27].
- Protein structure prediction: AI models are increasingly successful in predicting protein structures, which is crucial for the design of effective drugs, especially for proteins with challenging experimental characterization [29].
- Materials discovery: Deep learning can design biocompatible materials for drug delivery systems, enabling targeted delivery and controlled release, improving drug efficacy, and minimizing side effects [12].

3. New conceptual frameworks:

- New conceptual frameworks: Explainable AI: These are the models that provide information about the decision-making processes that are crucial for trust and collaboration between humans and AI. Allowing a better understanding of model predictions and guidance for future research [2].
- Quantum-assisted AI: Combining deep learning with quantum computing can revolutionize drug discovery. It is capable of mimicking complex biological processes with unprecedented precision, especially in systems that challenge classical computers [30].

4. Integration of omics data:
    - Comprehensive Data Analysis: DeepChem has already proven how it can be able to give efficient evaluation of large amounts of omics data that revolves around transcriptomics, proteomics, genomes, and metabolomics. The detailed technique can give us a clear understanding of how molecular mechanism works and its capability to be able to handle illness and medication responses [7].
    - Targeted and Personalized Drug Discovery: The strategy on how to develop drugs is significantly improved by the use of deep learning which gives us the window of opportunity to make vast analysis to build approaches that are more targeted and specified. We can make more individualized treatment methods for different kinds of patients with the discovery of intricate connections within huge biological datasets [4,14].
5. Robustness and Security:
    - Protective Measures Against Attacks: To avoid hostile attacks and unexpected effects it is very important to be able to maintain the durability and safety of deep learning models, such as DeepChem in this instance. Nowadays numerous research studies are focusing on creating strong and safe models that can fight off manipulation and hostile inputs that in the end will improve the accuracy and dependability of predictions [8].
    - Increased Confidence in Drug Discovery: There are two ways that strong models help build trust in the drug development process by protecting against possible flaws and also making the process safer. With these two important ways enforced, it makes sure that new therapeutic approaches are safe, and it gives everyone involved peace of mind that AI-driven drug creation processes are solid and reliable [8].

- Addressing Security Concerns: It is in truth very important for us to address security issues because when we are using artificial intelligence for the drug development process, it is inevitable for us to also input secret and private information. The security and integrity of vital data are safeguarded throughout the drug development lifecycle by using this approach, which strongly stresses the need for trustworthy and secure deep learning systems [8].

III. Ideas

- Can deep learning surpass human performance and knowledge?

  Deep learning excels in specific areas such as data analysis and pattern recognition, but is unlikely to surpass human performance and knowledge. AI lacks the human ability to take creative leaps, connect seemingly unconnected dots, and formulate intuitive hypotheses crucial to breakthrough discoveries. Human scientists bring years of experience, knowledge of specific biological systems, and understanding of disease context that AI models cannot fully replicate. This awareness of context is essential for interpreting data and guiding research directions. AI models are susceptible to biases and ethical dilemmas, requiring human oversight and guidance to ensure responsible development and application of drug discovery technologies, considering social implications and impacts. in health systems [12].

- Can AI explain the concept behind drug discovery?

  The application of AI in drug discovery is widespread, ranging from advanced image analysis, and prediction of molecular structure and function, to creating novel chemical entities. However, the underlying mathematical models are usually difficult to access and interpret by human beings. The decision-making process needs to be transparent, correct predictions for the wrong reasons need to be avoided, and unfair biases or unethical discriminations need to be averted.

  Currently, AI in drug discovery is in short of open-community platforms, where model interpretations and respective training data can be shared to collectively improve the software. MELLODDY (Machine Learning Ledger Orchestration for Drug Discovery) is an initiative taken towards the direction of

decentralization of model development and secured data handling between pharmaceutical companies. Further collaborations are needed to enable better model interpretability and foster trust between AI-driven hypotheses and human predictions [9].

IV. Investigations

1. Analysis

This project uses DeepChem as its main object of investigation. The analysis includes the current deep learning development for drug discovery using DeepChem, the strength of the object and its limitations, and challenges to overcome [19].

DeepChem is a Python library used for deep learning (DL) and machine learning (ML) on molecular and quantum datasets. On top of PyTorch and other well-known machine learning frameworks, it is constructed. It is intended to make developing and benchmarking new models as well as applying ML to new domains simple. It is also intended to facilitate the application of machine learning in production by offering user-friendly APIs for model export and deployment [19].

a. Current development of DeepChem

The goal of DeepChem is to offer an excellent open-source toolset that will enable anyone to employ deep learning in biology, materials science, quantum chemistry, and drug development. The biological sciences are currently experiencing a boom in the application of AI. Numerous recently funded AI-focused startups and activities at major biotech and pharmaceutical businesses are available. Following this advancement, DeepChem is used by professionals of the current industry, especially pharmacy. DeepChem offers assistance in the creation of potent medical therapies. Pharmacies may now contribute to the development of novel medication by building on the achievements of the past by using DeepChem.

Furthermore, DeepChem assists in lowering the price of medication as nowadays, creating new medical treatments is a highly skilled profession that only a select group of highly qualified professionals may engage in. DeepChem development can help

democratize new talents and increase competitiveness in drug discovery by contributing to open source technologies. The price of medicine may decrease with increased competition.

The current framework of DeepChem expresses neural networks for deep learning using scikit-learn and Google TensorFlow, an implementation for carrying out machine learning algorithms and an interface for expressing them [1]. Additionally, it uses the RDKit framework, a C++ toolkit for cheminformatics that is open-source and may be used with Python or Java, covering a range of common cheminformatics functions, such as fingerprinting, chemical reactions, coordinate generation (2D or 3D), molecular input/output, and substructure searches. The framework is used in Python to carry out simpler operations on molecular data, such as creating molecular graphs from SMILES strings. The Version 2.7.2 of the framework is currently in a public repository, in Github, while developing to add more TensorFlow models by using the graphics processing unit (GPU) for both training and inference [10].

b. Strengths of DeepChem

DeepChem puts user-friendliness first, in contrast to many other deep learning frameworks. Because of its user-friendly and comprehensive Python interface, scientists from a variety of backgrounds, including those without coding experiences can easily utilize it with the help of the DeepChem tutorial. Because of the modular design, the platform may be customized to meet particular research objectives with flexible workflows. Furthermore, the learning curve is accelerated by pre-built models and comprehensive tutorials, freeing researchers to concentrate on producing insights rather than overcoming technical challenges [23].

DeepChem also provides an extensive toolkit for a range of drug discovery applications. It can do a great deal of work, from simulating protein-ligand interactions and creating new drug candidates to predicting molecular characteristics and conducting virtual screening. This makes it possible for researchers to handle all phases of the discovery process on a single platform, saving time and effort by

removing the need to switch between several tools. Additionally, DeepChem is open-source, it encourages cooperation and quick development, continuously introducing new features and enhancing those that already exist [23].

c. Limitations of DeepChem and its challenges

Despite its strength from the ongoing development, there exist several broken features of DeepChem. First, DeepChem models can be black boxes, making it difficult to decipher the reasoning behind their predictions. This lack of easy interpretation can reduce the reliability of the results and limit their usefulness. Although DeepChem provides some tools to analyze the importance of features, further work is needed to improve the transparency and interpretability of models in scientific environments [23].

Second, DeepChem's main purpose is to be used on a single computer, where the input is fed through a single series transformation to produce the output, so it is useful for processing large data sets that require large amounts of computational power. Limited ability to handle sets and complex models. Although certain multiprocessing features are available, distributed training support and true multi-GPU support are still in the early stages of development. Therefore, its use may be limited to large-scale drug discovery efforts involving complex datasets or highly complex models [23].

Third, while DeepChem has a wide range of tools, it lacks native support for certain tasks, such as predicting ADMET properties (absorption, distribution, metabolism, excretion, toxicity) or designing proteins from scratch. Integrating these important drug discovery capabilities will expand the scope of DeepChem's applications and make its drug discovery platform more complete [23].

d. Applications in Drug Discovery:

Now we have discussed both the advantages and bottlenecks DeepChem provides, we will move on to provide its application in regards to drug discovery. As we know, the influence of DeepChem expands in many different aspects, which in turn shows how useful it is in stages of the drug development stages. We will discuss how it is

capable of simulating protein-ligand interactions. DeepChem can integrate itself in molecular docking simulations with algorithms it offers to be able to predict the best orientation and binding poses of ligands to target proteins [5].

As to why this simulation is important, it gives us an understanding of complicated ways in which molecules can interact with one another with different kinds of proteins, which in turn will give us more understanding of the potential of the strength of the interaction. Not only that, DeepChem's algorithm has the key metric that can predict and know the binding affinity that will give us the way to identify the compounds with high potential [5].

DeepChem's molecular modeling relies heavily on Quantitative Structure-Activity Relationship (QSAR) models, the reason why is that it is the most important part of the program. Not only will it aid us in the ays for predicting the activities of the new compounds, it also is able to facilitate the importance of molecules that will give good effect with the model's capability of establishing relationships between the chemical structure of compounds and their biological activities. It can give an advantage to researchers with its capability of contributing to the exploration of binding sites on target proteins. Researchers can influence their biological activities by designing molecules that can interact with specific protein regions [5].

DeepChem also can let us investigate and study more about the dynamic movements and interactions between proteins and ligands over time with its ability to facilitate molecular dynamics simulations. Which in the end proves how easy it is and practical it is to be implemented on different types of applications. Many successful studies have even shown how Deepchem molecular simulations accurately predicted binding modes and interactions that at the end led to the successful identification of new drugs [5].

Now after discussing how DeepChem relates to simulating protein-ligand interactions, we will move on to talk about its capacity to give us important insights into components that can influence a molecule's behavior in biological systems. One of the key components

DeepChem offers is the way it can measure a molecule's affinity for lipids or fats, in short lipophilicity. The explanation for why lipophilicity is important and has to be analyzed is how it can influence the drug's absorption, distribution, and metabolism inside our body. It can help us and researchers to evaluate the pharmaceutical kinetics information in regards to the compounds and can make the right decisions during drug development [5].

e. Community and Collaborative Development:

Rather than personal development in drug discovery, we must press the importance of how DeepChem globally creates an open-source framework that can provide a collaborative environment all around the world. Collaboration and teamwork globally, allow researchers and developers from different backgrounds to work together and contribute their expertise and innovations to DeepChem's platform. It will in the end speed up the growth and development of DeepChem which can give it the first choice for researchers to continually use on drug discoveries.

Since it enables users from all over the world to actively participate in forums and discussion groups, DeepChem's success is strongly dependent on the level of community participation. People are more encouraged to give their opinions or perspectives, collaborate to solve problems collectively and give ideas to the community. Collaborations like this enable us to enhance and improve its features for better functionalities. Not only that, the promptness of the platform's response to customer comments displays a dedication to integrating enhancements that are influenced by user perspectives. This ensures the capability of DeepChem to continue and adapt to the fastly growing and changing requirements that always persists in the scientific community. This serves as an example of how collaborative research might be the driving force behind the advancement of deep learning techniques, which are specifically developed for applications in the field of life sciences.

2. Design

This section shows the current framework of DeepChem and its methodology consisting of its data preprocessing, the molecular fingerprints of DeepChem. graph convolutions and dataset splitting.

a. Frameworks of DeepChem

The workflow for drug discovery activities such as solubility prediction is made simpler by DeepChem Framework. Same as other deep learning frameworks, DeepChem involves selecting and preparing data, crafting a model, training it on a dedicated set, rigorously evaluating its performance on unseen data, and finally, unleashing it to make predictions on novel inputs [18].

b. Data preprocessing

Worksets are the primary prerequisite for advancement in machine learning projects. DeepChem offers straightforward yet effective methods for handling massive volumes of data with ease. Designed to be easily integrable with various Python frameworks, like Tensorflow, Numpy, and Pandas. The dataset in DeepChem is a disk dataset where records are stored on the hard drive to access data efficiently. Mainly using the Numpy dataset, a dataset that contains all the data in a NumPy array, useful when processing small to large datasets that may contain in memory. There are also Image datasets that store all data in image files on disk, useful when working with models whose inputs and outputs are images [18].

Iterating across the dataset is the most effective approach to access the disk dataset. Users load small quantities of data at a time, process them, and then release random access memory (RAM) before adding more. To iterate over samples each time, the itersamples() method is utilized. A batch containing several samples is processed simultaneously by many DL models. To loop across these batches, iterbatches() is utilized. Alternatively, the to_dataframe() function can be used to obtain access to the dataset. Finally, the data is now copied into the DataFrame, a spreadsheet-like, two-dimensional data structure that is frequently used in programming and data analysis. Small datasets can also be used for each iteration of this method [18].

Procedure for processing:

itersamples()

```
In [5]: for X, y, w, id in test_dataset.itersamples():
            print(y, id)

[-1.70654087] C1c2ccccc2c3ccc4ccccc4c13
[0.2911162] COc1ccccc1Cl
[-1.42724759] COP(=S)(OC)Oc1cc(Cl)c(Br)cc1Cl
[-0.92546642] ClC(Cl)CC(=O)NC2=C(Cl)C(=O)c1ccccc1C2=O
[-1.95269767] ClC(Cl)C(c1ccc(Cl)cc1)c2ccc(Cl)cc2
[1.35148394] COC(=O)C=C
[-0.85919344] CN(C)C(=O)Nc2ccc(Oc1ccc(Cl)cc1)cc2
```

iterbatches()

```
In [6]: for X, y, w, ids in test_dataset.iterbatches(batch_size=50):
            print(y.shape)

(50, 1)
(50, 1)
(13, 1)
```

to_dataframe()

```
In [7]: test_dataset.to_dataframe()
```

Out[7]:

	X	y	w	ids
0	<deepchem.feat.mol_graphs.ConvMol object at 0x...	-1.706541	1.0	C1c2ccccc2c3ccc4ccccc4c13
1	<deepchem.feat.mol_graphs.ConvMol object at 0x...	0.291116	1.0	COc1ccccc1Cl
2	<deepchem.feat.mol_graphs.ConvMol object at 0x...	-1.427248	1.0	COP(=S)(OC)Oc1cc(Cl)c(Br)cc1Cl
3	<deepchem.feat.mol_graphs.ConvMol object at 0x...	-0.925466	1.0	ClC(Cl)CC(=O)NC2=C(Cl)C(=O)c1ccccc1C2=O
4	<deepchem.feat.mol_graphs.ConvMol object at 0x...	-1.952698	1.0	ClC(Cl)C(c1ccc(Cl)cc1)c2ccc(Cl)cc2
...	...	...	...	...
108	<deepchem.feat.mol_graphs.ConvMol object at 0x...	0.646150	1.0	FC(F)(F)C(Cl)Br
109	<deepchem.feat.mol_graphs.ConvMol object at 0x...	1.505805	1.0	CNC(=O)ON=C(SC)C(=O)N(C)C
110	<deepchem.feat.mol_graphs.ConvMol object at 0x...	-0.007586	1.0	CCSCCSP(=S)(OC)OC
111	<deepchem.feat.mol_graphs.ConvMol object at 0x...	-0.049716	1.0	CCC(C)C
112	<deepchem.feat.mol_graphs.ConvMol object at 0x...	-0.684990	1.0	COP(=O)(OC)OC(=CC(c1cc(Cl)c(Cl)cc1Cl)c1Cl

113 rows × 4 columns

Aside from accessing the prepared data from DeepChem, it is possible to create a user designed dataset using Numpy by passing the data using an array to the constructor and wrapping them in NumpyDataset [18].

Procedure for user designed dataset:

```
In [8]: import numpy as np

        X = np.random.random((10, 5))
        y = np.random.random((10, 2))
        dataset = dc.data.NumpyDataset(X=X, y=y)
        print(dataset)

<NumpyDataset X.shape: (10, 5), y.shape: (10, 2), w.shape: (10, 1), ids: [0 1 2 3 4 5 6 7 8 9], task_names:
[0 1]>
```

```
In [9]: dataset.to_dataframe()
```

Out[9]:

	X1	X2	X3	X4	X5	y1	y2	w	ids
0	0.547330	0.919941	0.289138	0.431806	0.776672	0.532579	0.443258	1.0	0
1	0.980867	0.642487	0.460640	0.500153	0.014848	0.678259	0.274029	1.0	1
2	0.953254	0.704446	0.857458	0.378372	0.705789	0.704786	0.901080	1.0	2
3	0.904970	0.729710	0.304247	0.861546	0.917029	0.121747	0.758845	1.0	3
4	0.464144	0.059168	0.600405	0.880529	0.688043	0.595495	0.719861	1.0	4
5	0.820482	0.139002	0.627421	0.129399	0.920024	0.634030	0.464525	1.0	5
6	0.113727	0.551801	0.536189	0.066091	0.311320	0.699331	0.171532	1.0	6
7	0.516131	0.918503	0.429036	0.844973	0.639367	0.464089	0.337989	1.0	7
8	0.809393	0.201450	0.821420	0.841390	0.100026	0.230462	0.376151	1.0	8
9	0.076750	0.389277	0.350371	0.291806	0.127522	0.544606	0.306578	1.0	9

c. Molecular fingerprints

In molecular biology and chemistry, compact representations of complicated data are referred to as fingerprints in the context of deep learning. A fingerprint is a fixed-length array in which distinct entries signify the existence of various molecular characteristics. Similar fingerprints show that two molecules have numerous characteristics, which suggests that their chemistry is similar [17].

A type of fingerprint known as a molecular fingerprint functions well for tiny drugs, such as molecules. "Extended Connectivity Fingerprint," or "ECFP" is the molecular fingerprint offered by DeepChem [17].

Atoms are first categorized by the ECFP algorithm only on the basis of their immediate characteristics and bonds. Every distinct design is a feature. It then looks at larger circular areas repeatedly, finding new features. A higher level feature is created when two specified features are bound together, and the matching element is set for any molecule that includes the higher level feature [17].

Example:

The training model begins with a fixed length array. Fully connected layers are stacked to create a MultiTaskClassifier.

```
In [5]: import numpy as np

        model.fit(train_dataset, nb_epoch=10)
        metric = dc.metrics.Metric(dc.metrics.roc_auc_score)
        print('training set score:', model.evaluate(train_dataset, [metric], transformers))
        print('test set score:', model.evaluate(test_dataset, [metric], transformers))

        training set score: {'roc_auc_score': 0.9550063508563469}
        test set score: {'roc_auc_score': 0.7781819573695475}
```

d. Graph convolutions

The most potent deep learning instruments for handling molecular data are graph convolutions. This is because molecules are naturally thought of as graphs [3].

Graph convolutions are similar to convolutional neural networks (CNN) is a network inspired by the human visual cortex, automatically extracts features from data like images by utilizing shared weights and local connections, simplifying training and boosting efficiency. The input is a pixel grid. Every pixel has a vector of data values, such as the color channels for red, green, and blue. A sequence of convolutional layers process the data. To create a new data vector for a pixel, each layer combines the data from the pixel and its neighbors. While subsequent layers identify broader, more abstract patterns, earlier levels identify local, small-scale patterns. Convolutional layers frequently switch places with pooling layers to carry out operations like max or min over local regions [3].

Instead of a network, graph convolutions are graphs. Convolutional and pooling layers mix data from connected nodes to create a new data vector for each node, starting with a data vector for each node in the network [3].

MoleculeNet (data source provided by DeepChem) is used to load the dataset and train the GraphConvModel. Following the dataset's training, the model's performance is assessed. Model performance is represented by a specific metric and metrics. The ROC-AUC score is used in the example. Utilizing the model.evaluate() function, one may verify the correctness and score [3].

Example:

e. Dataset splitting

Dividing data into validation, training and test sets is important in deep learning. A Splitter object defines a method in DeepChem for dividing samples into different datasets. Selecting the right approach for

your data is crucial. If not, the performance of the trained model is not at its peak [6].

Because chemical datasets frequently contain a large number of molecules that are strikingly similar to one another, this has significant implications for those datasets. A simplistic partitioning of the data into training and test sets will result in a large number of molecules in the training set that are strikingly similar to the molecules in the test set, not differentiating their differences [6].

- Random splitter

  It uses a random process to choose samples for the test, validation, and training sets.

```
splitter: random
training set score: {'roc_auc_score': 0.9560766203173238}
test set score: {'roc_auc_score': 0.8088861019955839}
```

- Random stratified splitter

  Distributing equally between the positive and negative samples.

- Scaffold splitter

  It determines which scaffold makes up each molecule's core and makes sure that molecules with the same scaffold are included in the same dataset.

```
splitter: scaffold
training set score: {'roc_auc_score': 0.9582835670901536}
test set score: {'roc_auc_score': 0.6803307954037949}
```

- Butina splitter

  Datasets are grouped according to their molecular fingerprints, which means that individuals with comparable fingerprints will typically be found in the same dataset. This splitting technique works best with small to medium large datasets since its time complexity scales as the square of the number of molecules.

```
splitter: butina
training set score: {'roc_auc_score': 0.9578120869103354}
test set score: {'roc_auc_score': 0.6057007877463954}
```

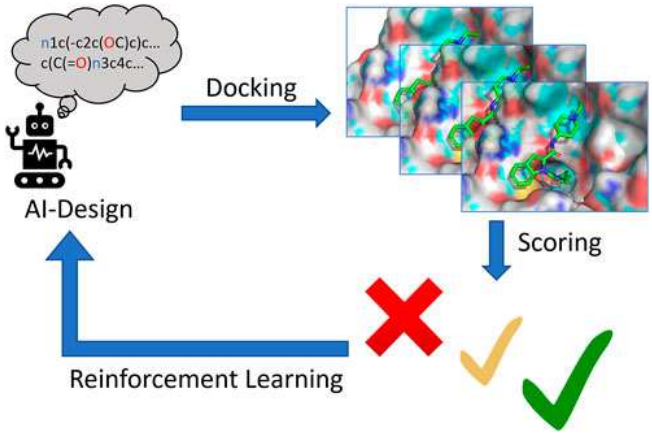
- Specified splitter

  The splitting mechanism is set up by the user.

3. Experiments

   3.1 Protein ligand interaction modeling

The first experiment observation is focused on protein ligand interaction modeling, discussing the experiment dataset, setup, and tasks. Additionally, performance modification and results are attached.



Molecular docking is a computational method used to predict how well a small molecule (ligand) binds to a protein. Molecular docking is a critical tool in drug development because it can practically predict how prospective drug molecules would fit and interact with target proteins. It streamlines the whole drug development process from target identification to lead optimization by assisting in the identification of potentially active compounds, maximizing their binding strength and leading for potential new drugs [16].

A ligand is a tiny molecule that interacts with a protein, typically non-covalently. By using geometric calculations, molecular docking determines a binding pose for a small molecule interacting with a protein in an appropriate binding pocket, a section of the protein that has a groove where the small molecule can rest.

The experiment starts by loading a protein-ligand complex dataset (PDBbind), performing programmatic molecular docking, featurizing protein-ligand complexes with interaction fingerprints, and fitting a random forest model and predicting binding affinities [16].

a. Loading protein ligand complex dataset

For the ligand and protein targets from PDBbind, a CSV containing SMILES strings of ligands as well as PDB files are utilized.

```
data_dir = dc.utils.get_data_dir()
dataset_file = os.path.join(data_dir, "pdbbind_core_df.csv.gz")

if not os.path.exists(dataset_file):
  print('File does not exist. Downloading file...')
  download_url("https://s3-us-west-1.amazonaws.com/deepchem.io/datasets/pdbbind_core_df.csv.gz")
  print('File downloaded...')

raw_dataset = load_from_disk(dataset_file)
raw_dataset = raw_dataset[['pdb_id', 'smiles', 'label']]
```

Raw data:

```
In [10]: raw_dataset.head(2)
```

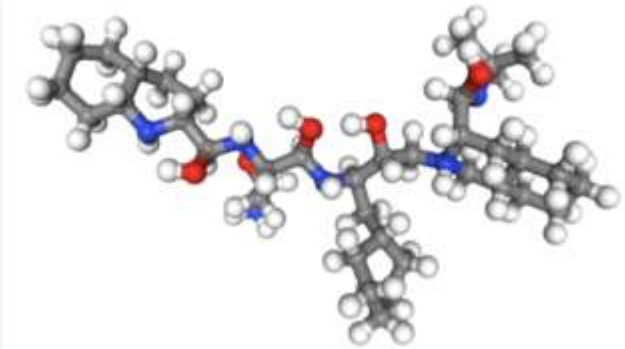
Out[10]:

	pdb_id	smiles	label
0	2d3u	CC1CCCCC1S(O)(O)NC1CC(C2CCC(CN)CC2)SC1C(O)O	6.92
1	3cyx	CC(C)(C)NC(O)C1CC2CCCCC2C[NH+]1CC(O)C(CC1CCCC...	8.00

Use MDTraj and nglview to visualize proteins and ligands:

```
In [17]: display(v) # interactive view outside Colab
```

```
NGLWidget()
```



```
In [18]: view = nglview.show_mdtraj(protein_mdtraj)
         display(view) # interactive view outside Colab
```

```
NGLWidget()
```

b. Performing programmatic molecular docking

The binding affinities between protein ligand complexes are estimated using molecular docking. A ligand in a binding pocket can be generated into poses (geometric configurations) by following three steps in the procedure: 1) identifying binding pockets in the target protein; 2) creating poses; and 3) score a

pose. The score indicates that the primary goal is to find potential ligands that have a strong interaction with a target protein [25].

Identify pockets:

```
In [19]: finder = dc.dock.binding_pocket.ConvexHullPocketFinder()
         pockets = finder.find_pockets('3cyx.pdb')
         len(pockets)  # number of identified pockets

Out[19]: 36
```

Creating poses:

```
vpg = dc.dock.pose_generation.VinaPoseGenerator()
```

```
In [21]: %%time
         complexes, scores = vpg.generate_poses(molecular_complex=('3cyx.pdb', 'ligand_3cyx.pdb'),  # protein-ligand files for docking
                                                out_dir='vina_test',
                                                generate_scores=True
                                                )
```

Score the pose:

```
In [22]: scores

Out[22]: [-9.484, -9.405, -9.195, -9.151, -8.9, -8.696, -8.687, -8.633, -8.557]
```

c. Featurizing protein ligand complexes with interaction fingerprints

Because DeepChem does not have specific protein-ligand complex fingerprints, it makes use of manually selected feature generators such as CircularFingerprint and ContactCircularFingerprint. These transform intricate structures into vectors that deep learning algorithms may use.

```
In [33]: fp_featurizer = dc.feat.CircularFingerprint(size=2048)

In [34]: features = fp_featurizer.featurize([Chem.MolFromPDBFile(l) for l in ligands])

In [35]: dataset = dc.data.NumpyDataset(X=features, y=labels, ids=pdbids)
         train_dataset, test_dataset = dc.splits.RandomSplitter().train_test_split(dataset, seed=42)
```

After the featurizing is finished, the model is ready for learning.

d. Fitting a random forest model

A supplied model class first is initialized in order to fit a deepchem model. In this instance, the object has developed a convenience class that can be used to interface with deepchem by wrapping over any model that is accessible in Sci-Kit Learn [25].

task_types, model_params, and a model_instance indicating RandomForestRegressor, in order to create a SklearnModel.

```
In [38]: seed = 42 # Set a random seed to get stable results
         sklearn_model = RandomForestRegressor(n_estimators=100, max_features='sqrt')
         sklearn_model.random_state = seed
         model = dc.models.SklearnModel(sklearn_model)
         model.fit(train_dataset)
```

e. Predicting binding affinities

To predict the binding affinities, the experiment uses an evaluator. However, predicting the affinities through the provided data is hard. R^2 value for the test set shows that there are no significant outputs from the model [25].

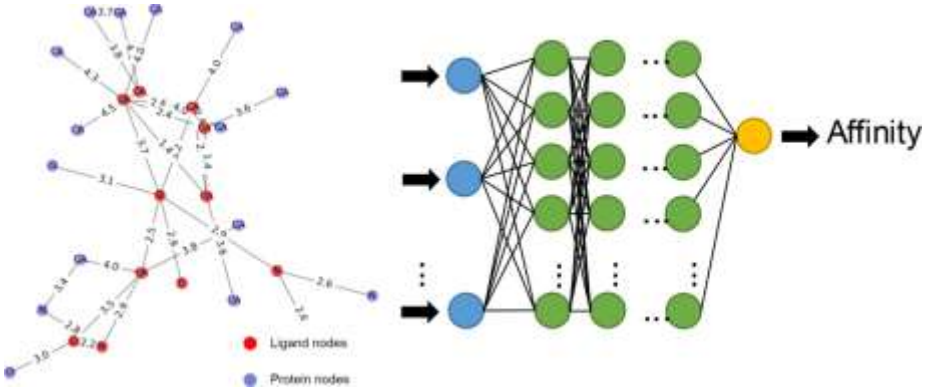
```
In [39]: # use Pearson correlation so metrics are > 0
         metric = dc.metrics.Metric(dc.metrics.pearson_r2_score)

         evaluator = Evaluator(model, train_dataset, [])
         train_r2score = evaluator.compute_model_performance([metric])
         print("RF Train set R^2 %f" % (train_r2score["pearson_r2_score"]))

         evaluator = Evaluator(model, test_dataset, [])
         test_r2score = evaluator.compute_model_performance([metric])
         print("RF Test set R^2 %f" % (test_r2score["pearson_r2_score"]))

         RF Train set R^2 0.888697
         RF Test set R^2 0.007797
```

```
In [40]: # Compare predicted and true values
         list(zip(model.predict(train_dataset), train_dataset.y))[:5]

Out[40]: [(6.862549999999994, 7.4),
          (6.616400000000008, 6.85),
          (4.852004999999995, 3.4),
          (6.43060000000001, 6.72),
          (8.66322999999999, 11.06)]

In [41]: list(zip(model.predict(test_dataset), test_dataset.y))[:5]

Out[41]: [(5.960549999999999, 4.21),
          (6.051305714285715, 8.7),
          (5.799900000000003, 6.39),
          (6.433881666666665, 4.94),
          (6.7465399999999995, 9.21)]
```

Result analysis:

The result of this experiment is not satisfying because the model does not show significant output from the learning, as indicated by R^2. To create a more useful model, a better data cleaning and docking

strategy is needed. Additionally, extra information that is the protein data should be added to increase the efficiency of the model.



3.2 Protein deep learning

The second experiment we will focus on protein deep learning, where we experiment and analyze on a comprehensive investigation of diverse techniques for featurizing protein sequences. In this experiment we will use methodologies like one-hot encoders and amino acid composition, using DeepChem's tools. the goal is to create a model that can predict and estimate the melting temperature of proteins.

And in regards of the database, we will cite the (https://aip.scitation.org/doi/10.1063/1.4947493) that has many thermodynamic information about proteins. And also we will have insights about the change in Gibbs Free Energy (ΔΔG°) [15].

a. Setup

Setup and add installation commands.

```
(base) ceciliaalexa@cecilias-MacBook-Pro ~ % curl -Lo conda_installer.py https://raw.githubusercontent.com/deepchem/deepchem/master/scripts/colab_install.py
import conda_installer
conda_installer.install()
!/root/miniconda/bin/conda info -e
```

```
(base) ceciliaalexa@cecilias-MacBook-Pro ~ % !pip install --pre deepchem
!pip install propy3
```

b. Data Extraction

Retrieve the dataset featured in the publication from the DeepChem.

```
import deepchem as dc
import os
from deepchem.utils import download_url
data_dir = dc.utils.get_data_dir()
download_url("https://deepchemdata.s3-us-west-1.amazonaws.com/datasets/pucci-proteins-appendistable1
    .csv",dest_dir=data_dir)
print('Dataset Downloaded at {}'.format(data_dir))
dataset_file = os.path.join(data_dir, 'pucci-proteins-appendistable1.csv')

```

Output the csv data file with this code.

```
import pandas as pd
data = pd.read_csv(dataset_file)
data

```

Output:

	Unnamed: 0	N	PDBid	Chain	RESN	RESwt	RESmut	ΔTmexp	Tmexp [wt]	ΔΔHmexp
0	NaN	1	1aky	A	8	VAL	ILE	-1.8	47.8	70
1	NaN	2	1aky	A	46	GLN	GLU	-1.3	47.6	60
2	NaN	3	1aky	A	77	THR	HIS	-1.1	47.6	130
3	NaN	4	1aky	A	110	THR	HIS	-4.8	47.6	165
4	NaN	5	1aky	A	169	ASN	ASP	-0.6	47.6	140
...	...	...	...	...	...	...	...	...	...	...
1621	NaN	1622	6pti_m52l	A	15	LYS	SER	-1.3	91.7	-5
1622	NaN	1623	6pti_m52l	A	15	LYS	THR	-1.1	91.7	-9
1623	NaN	1624	6pti_m52l	A	15	LYS	VAL	-6.3	91.7	4
1624	NaN	1625	6pti_m52l	A	15	LYS	TRP	-7.5	91.7	17
1625	NaN	1626	6pti_m52l	A	15	LYS	TYR	-6.6	91.7	4

1626 rows × 23 columns

...	ΔΔGexp(T)	T	Nres	R (Å)	Protein	Organism	Ref.	pH	Exp.Tech.	
...	5.0	25	220	1.63	ADK	Yeast	[1]	[7.5]	FL	NaN
...	4.0	25	220	1.63	ADK	Yeast	[1]	[7.7]	FL	NaN
...	8.0	25	220	1.63	ADK	Yeast	[1]	[7.8]	FL	NaN
...	11.0	25	220	1.63	ADK	Yeast	[1]	[7.6]	FL	NaN
...	9.0	25	220	1.63	ADK	Yeast	[1]	[7.6]	FL	NaN
...	...	...	...	...	...	...	...	...	...	...
...	1.2	25	58	1.00	PTI M52L	Bovine	[232]	[3.0]	DSC	NaN
...	-3.6	25	58	1.00	PTI M52L	Bovine	[232]	[3.0]	DSC	NaN
...	4.7	25	58	1.00	PTI M52L	Bovine	[232]	[3.0]	DSC	NaN
...	8.5	25	58	1.00	PTI M52L	Bovine	[232]	[2.5]	DSC	NaN
...	4.6	25	58	1.00	PTI M52L	Bovine	[232]	[3.0]	DSC	NaN

Extract a small DataFrame that only contains a unique PDBid code and its respective melting temperature.

```
WT_Tm = data[['PDBid','Tmexp [wt]']]
WT_Tm.set_index('PDBid',inplace=True)
WT_Tm

```

PDBid	Tmexp [wt]
1aky	47.6
1aky	47.6
1aky	47.6
1aky	47.6
1aky	47.6
...	...
5pti_m52l	91.7
5pti_m52l	91.7
5pti_m52l	91.7
5pti_m52l	91.7
5pti_m52l	91.7

1626 rows × 1 columns

Create a dictionary that contains as keys, the pdbid of each protein and as values, the wild type melting temperature

```
dict_WT_TM = {}
for k,v in WT_Tm.itertuples():
    if(k not in dict_WT_TM):
        dict_WT_TM[k]=float(v)

pdbs = data[data['PDBid'].str.len()<5]
pdbs = pdbs[pdbs['Chain'] == "A"]

pdbs[['RESN','RESwt','RESmut']]
```

	RESN	RESwt	RESmut
0	8	VAL	ILE
1	48	GLN	GLU
2	77	THR	HIS
3	110	THR	HIS
4	169	ASN	ASP
...	...	...	...
1604	36	GLY	ALA
1605	36	GLY	SER
1606	37	GLY	ALA
1607	39	ARG	ALA
1608	46	LYS	ALA

1509 rows × 3 columns

This piece of code gets information about the total amount of mutations and changes in the melting temperature (MT). In addition, residue changes are turned into a one-letter code using a database.

```
alls=[]
for resnum,wt in pdbs[['RESN','RESwt','RESmut','PDBid','ΔTmexp']].iteritems():
    alls.append(wt.values)
d = {'CYS': 'C', 'ASP': 'D', 'SER': 'S', 'GLN': 'Q', 'LYS': 'K',
     'ILE': 'I', 'PRO': 'P', 'THR': 'T', 'PHE': 'F', 'ASN': 'N',
     'GLY': 'G', 'HIS': 'H', 'LEU': 'L', 'ARG': 'R', 'TRP': 'W',
     'ALA': 'A', 'VAL':'V', 'GLU': 'E', 'TYR': 'Y', 'MET': 'M'}
resnum=alls[0]
wt=[d[x.strip()] for x in alls[1]] # extract the Wildtype aminoacid with one letter code
mut=[d[x.strip()] for x in alls[2]] # extract the Mutation aminoacid with one letter code
codes=alls[3] # PDB code
tms=alls[4] # Melting temperature

-AA {}, Resnum {}, MUT-AA {},DeltaTm {}".format(codes[0],wt[0],resnum[0],mut[0],tms[0]))
```

```
pdbid 1aky, WT-AA V, Resnum 8, MUT-AA I,DeltaTm -1.5
```

c. PDB download

Downloading the pdbs by PDBID using the pdbfixer tool.

```
(base) ceciliaalexa@cecilias-MacBook-Pro ~ % from pdbfixer import PDBFixer
from simtk.openmm.app import PDBFile
```

```
(base) ceciliaalexa@cecilias-MacBook-Pro ~ % !mkdir PDBs
```

```
import os
import time
t0 = time.time()

downloaded = os.listdir("PDBs")
PDBs_ids= set(pdbs['PDBid'])
pdb_list = []
print("Start Download ")
for pdbid in PDBs_ids:
    name=pdbid+".pdb"
    if(name in downloaded):
        continue
    try:
        fixer = PDBFixer(pdbid=pdbid)
        fixer.findMissingResidues()
        fixer.findNonstandardResidues()
        fixer.replaceNonstandardResidues()
        fixer.removeHeterogens(True)
        fixer.findMissingAtoms()
        fixer.addMissingAtoms()
        PDBFile.writeFile(fixer.topology, fixer.positions, open('./PDBs/%s.pdb' % (pdbid),
            'w'),keepIds=True)
    except:
        print("Problem with {}".format(pdbid))
print("Total Time {}".format(time.time()-t0))
```

We are able to extract crucial information from the mutation string, including the wild-type amino acid, position, and the desired mutation. this will help identifying a sequence of aminoacids base on PDB structures

```
from Bio.PDB.PDBParser import PDBParser
from Bio.PDB.Polypeptide import PPBuilder
ppb=PPBuilder()
def GetSeqFromPDB(pdbid):
    structure = PDBParser().get_structure(pdbid.split(".")[0], 'PDBs/{}'.format(pdbid))
    seqs=[]
    return ppb.build_peptides(structure)
```

```
import warnings; warnings.simplefilter('ignore')
test="1ezm"
print(test)
seq = GetSeqFromPDB("{}.pdb".format(test))[0].get_sequence()
print("Original Sequence")
print(seq)

```

now let us see the output of the original sequence

```
1ezm
Original Sequence
AEAGGPGGNQKIGKYTYGSDYGPLIVNDRCEMDDGNVITVDMNSSTDDSKTTPFRFACPTNTYKQVNGAYSPLNDAHFFGGVVFKLYRDW
FGTSPLTHKLYMKVHYGRSVENAYWDGTAMLFGDGATMFYPLVSLDVAAHEVSHGFTEQNSGLIYRGQSGGMNEAFSDMAGEAAEFYMRG
KNDFLIGYDIKKGSGALRYMDQPSRDGRSIDNASQYYNGIDVHHSSGVYNRAFYLLANSPGWDTRKAFEVFVDANRYYWTATSNYNSGAC
GVIRSAQNRNYSAADVTRAFSTVGVTCPSAL
```

Now programming the code about the information in regards of the mutation.

```
informSeq=GetSeqFromPDB(test+".pdb")[0].__repr__()
print("Seq information",informSeq)
start = re.findall('[0-9]+',informSeq)[0]
print("Reported Mutation {}{}{}".format("R",179,"A"))
numf =179 - int(start) + 1  # fix some cases of negative aminoacid numbers
```

the output is

```
Seq information <Polypeptide start=1 end=301>
Reported Mutation R179A
```

Now we shall to move on to create a code for mutation in the sequence.

```
mutfinal = "R{}A".format(numf)
print("Real Mutation = ",mutfinal)
mutseq = MutateSeq(seq,mutfinal)
print(mutseq)
```

The output is

```
Real Mutation =  R179A
AEAGGPGGNQKIGKYTYGSDYGPLIVNDRCEMDDGNVITVDMNSSTDDSKTTPFRFACPTNTYKQVNGAYSPLNDAHFFGGVVFKLYRDW
FGTSPLTHKLYMKVHYGRSVENAYWDGTAMLFGDGATMFYPLVSLDVAAHEVSHGFTEQNSGLIYRGQSGGMNEAFSDMAGEAAEFYMAG
KNDFLIGYDIKKGSGALRYMDQPSRDGRSIDNASQYYNGIDVHHSSGVYNRAFYLLANSPGWDTRKAFEVFVDANRYYWTATSNYNSGAC
GVIRSAQNRNYSAADVTRAFSTVGVTCPSAL
```

We now extract relevant information from protein structures represented in the PDB format. The resulting mutated sequence and the corresponding change in melting temperature (ΔTm) are then stored in the 'information' dictionary. So in situations where mutations encounter issues, for example being out of the protein sequence range or involving mismatched wild-type amino acids, we log these instances in the 'failures' list.

```
information = {}
count = 1
failures=[]
for code,tm,numr,wt_val,mut_val in zip(codes,tms,resnum,wt,mut):
    count += 1
    seq = GetSeqFromPDB("{}.pdb".format(code))[0].get_sequence()
    mutfinal="WT"
    if("{}-{}".format(code,mutfinal) not in information):
        informSeq=GetSeqFromPDB(code+".pdb")[0].__repr__()
        start = re.findall('[-0-9]+',informSeq)[0]
    if(int(start)<0):
        numf =numr - int(start) # if start is negative 0 is not used as resnumber
    else:
        numf =numr - int(start) + 1
    mutfinal = "{}{}{}".format(wt_val,numf,mut_val)
    mutseq = MutateSeq(seq,mutfinal)
    if(mutseq==None):
        failures.append((code,mutfinal))
        continue
    information["{}-{}".format(code,mutfinal)]=[mutseq,dict_WT_TM[code]-float(tm)]
```

d. Deep Learning and Machine Learning Models using proteins sequences

We now use DeepChem to encode protein sequences and their corresponding ΔTm values into a format suitable for machine learning analysis. First, we compile the mutated sequences and their ΔTm values into separate lists, seq_list and deltaTm, respectively, from the information stored in the earlier processing steps. Also by using the OneHotFeaturizer from DeepChem, we convert the sequences into one-hot encoded vectors, considering a predefined amino acid code set and accommodating the determined maximum sequence length. This ensures consistent input dimensions for subsequent modeling. Finally, we flatten the features to create a vector representation for each sequence and assemble the dataset using DeepChem's NumpyDataset. This dataset, denoted as dc_dataset, encapsulates the one-hot encoded sequence vectors and their corresponding ΔTm values, laying the groundwork for training and evaluating our machine learning model.

```
import deepchem as dc
import tensorflow as tf

seq_list=[]
deltaTm=[]
for i in information.values():
    seq_list.append(i[0])
    deltaTm.append(i[1])
max_seq= 0
for i in seq_list:
    if(len(i)>max_seq):
        max_seq=len(i)

codes = ['A', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L',
         'M', 'N', 'P', 'Q', 'R', 'S', 'T', 'V', 'W', 'Y']
OneHotFeaturizer = dc.feat.OneHotFeaturizer(codes,max_length=max_seq)
features = OneHotFeaturizer.featurize(seq_list)
features_vector  = []
for i in range(len(features)):
    features_vector.append(features[i].flatten())

dc_dataset = dc.data.NumpyDataset(X=features_vector,y=deltaTm)

dc_dataset
```

And the output for the program is:

```
<NumpyDataset X.shape: (1497, 13188), y.shape: (1497,), w.shape: (1497,), task_names: [0]>
```

Our last step is to use DeepChem's Keras API to test our machine learning model's ability to guess Δ values. The Mean Absolute Error is the loss function we choose for training, and the Adam optimizer is used to find the best solution. The start the training process we use the fit method and select the batch size, training datasets, and validation datasets. The model learns how to keep the MAE loss between expected and real ΔTm values as low as possible during training. After training, we can use matplotlib to make a plot that shows the training and validation loss trends over time. This gives us a visual picture of how well the model generalises.

```
from deepchem import splits
splitter = splits.RandomSplitter()
train, test  = splitter.train_test_split(dc_dataset,seed=42)
import tensorflow.keras as keras
#from  keras import layers
model = tf.keras.Sequential([
    keras.layers.Dense(units=32, activation='relu', input_shape=dc_dataset.X.shape[1:]),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(units=32, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(units=1),
])

model.compile(loss='mae', optimizer='adam')

print(model.summary())

history = model.fit(
    train.X, train.y,
    validation_data=(test.X,test.y),
    batch_size=100,
    epochs=30,
)

## perform a plot of loss vs epochs
import matplotlib.pyplot as plt
history_df = pd.DataFrame(history.history)
history_df[['loss', 'val_loss']].plot()
```

the output is

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_6 (Dense)              (None, 32)                422048
_________________________________________________________________
dropout_4 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_7 (Dense)              (None, 32)                1056
_________________________________________________________________
dropout_5 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 33
=================================================================
Total params: 423,137
Trainable params: 423,137
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/30
12/12 [==============================] - 1s 20ms/step - loss: 62.4284 -
val_loss: 51.3894
Epoch 2/30
12/12 [==============================] - 0s 10ms/step - loss: 45.6483 -
val_loss: 22.2757
Epoch 3/30
12/12 [==============================] - 0s 12ms/step - loss: 19.4803 -
val_loss: 11.7996
Epoch 4/30
12/12 [==============================] - 0s 11ms/step - loss: 18.0858 -
val_loss: 8.6031
Epoch 5/30
12/12 [==============================] - 0s 11ms/step - loss: 14.5904 -
val_loss: 9.4577
Epoch 6/30
12/12 [==============================] - 0s 10ms/step - loss: 14.3444 -
val_loss: 7.3325
Epoch 7/30
12/12 [==============================] - 0s 10ms/step - loss: 13.6787 -
val_loss: 7.5799
Epoch 8/30
12/12 [==============================] - 0s 10ms/step - loss: 14.0674 -
val_loss: 6.6186
Epoch 9/30
12/12 [==============================] - 0s 11ms/step - loss: 12.8215 -
val_loss: 7.4920
Epoch 10/30
12/12 [==============================] - 0s 11ms/step - loss: 13.0748 -
val_loss: 5.4614
Epoch 11/30
12/12 [==============================] - 0s 11ms/step - loss: 12.3646 -
val_loss: 6.7943
Epoch 12/30
12/12 [==============================] - 0s 10ms/step - loss: 11.5250 -
val_loss: 5.2098
Epoch 13/30
```

```
12/12 [==============================] - 0s 11ms/step - loss: 11.4129 -
val_loss: 5.6157
Epoch 16/30
12/12 [==============================] - 0s 11ms/step - loss: 11.4406 -
val_loss: 6.0425
Epoch 17/30
12/12 [==============================] - 0s 12ms/step - loss: 11.4762 -
val_loss: 5.3330
Epoch 18/30
12/12 [==============================] - 0s 11ms/step - loss: 11.4820 -
val_loss: 8.5933
Epoch 19/30
12/12 [==============================] - 0s 11ms/step - loss: 11.1607 -
val_loss: 5.6460
Epoch 20/30
12/12 [==============================] - 0s 12ms/step - loss: 11.2637 -
val_loss: 5.3362
Epoch 21/30
12/12 [==============================] - 0s 11ms/step - loss: 11.4390 -
val_loss: 6.2368
Epoch 22/30
12/12 [==============================] - 0s 10ms/step - loss: 10.8070 -
val_loss: 6.2875
Epoch 23/30
12/12 [==============================] - 0s 11ms/step - loss: 11.1277 -
val_loss: 5.3496
Epoch 24/30
12/12 [==============================] - 0s 10ms/step - loss: 11.1626 -
val_loss: 5.0670
Epoch 25/30
12/12 [==============================] - 0s 10ms/step - loss: 10.9386 -
val_loss: 5.0440
Epoch 26/30
12/12 [==============================] - 0s 14ms/step - loss: 11.0402 -
val_loss: 5.6448
Epoch 27/30
12/12 [==============================] - 0s 14ms/step - loss: 10.7278 -
val_loss: 5.3868
Epoch 28/30
12/12 [==============================] - 0s 13ms/step - loss: 10.6073 -
val_loss: 5.4149
Epoch 29/30
12/12 [==============================] - 0s 12ms/step - loss: 11.0812 -
val_loss: 5.9349
Epoch 30/30
12/12 [==============================] - 0s 14ms/step - loss: 10.2336 -
val_loss: 5.7489
```

e. DeepChem Keras Model

```
model_dc = dc.models.KerasModel(model, dc.models.losses.L1Loss())
model_dc.fit(train)
```

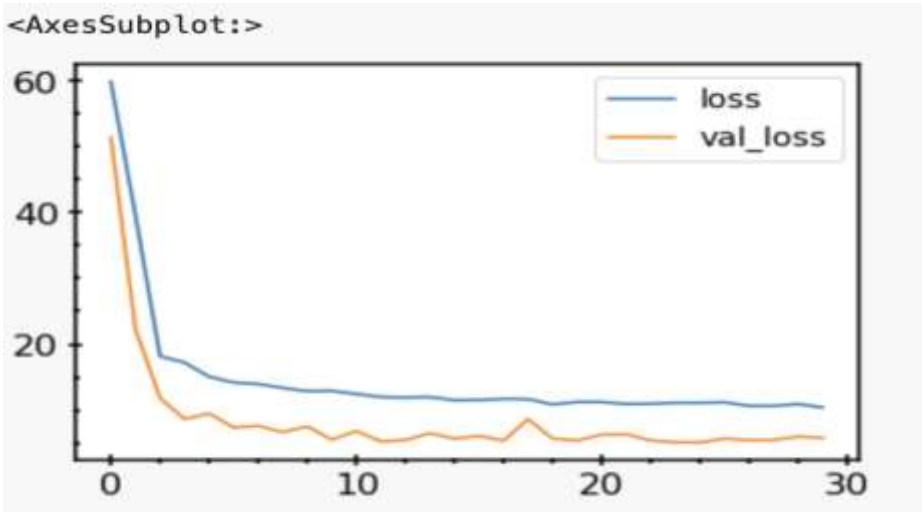
The output is

```
10.399123382568359
```

Now we will visualize the graph

```
History_df = pd.DataFrame(model_dc.model.history.history)
History_df[['loss', 'val_loss']].plot()
```

The output is



f. Result Analysis:

The result of this experiment showed a satisfying result that accurately visualize the correlation of predicted and actual ΔTm values [15].

V. Conclusions and suggestions

Artificial neural networks are being used to find complex connections in very large datasets. With the help of deep learning in drug discovery has changed the way things are done traditionally. The ability of these networks to model complex relationships shows how much they have changed things. The introduction sets the scene by talking about how deep learning, especially tools like DeepChem, makes computer-assisted drug finding possible. The focus on modeling complicated connections shows how it could open up new areas of biology knowledge, which could lead to more focused and effective drug development.

In regards to AI in drug development, there are two opposite spectrums that it brings to the table, with the promises that extraordinary it also brings us to many critical issues that need to be paid more attention to. Diverse and high-quality datasets are known to be necessary for making models that can be trusted, but they need to be handled carefully because they could be biased and because they could make social problems worse. We also have to bring awareness on how important it is to set human control and careful thought about how technology will affect society, as it is important to find a balance between doing the right thing and making progress in technology. Realizing how and where these problems show a dedication to responsible AI use, making sure that the advantages of drug finding are shared fairly and are easy to get.

Moreover, it enhances the overall availability of medical care by reducing expenses and simplifying the acquisition of drugs. To showcase the commitment to enhancing DeepChem, it is essential to possess a deep understanding of its constraints, including those related to scalability and the interpretability of models. The research and development endeavors demonstrate an adaptable and collaborative strategy that appreciates the synergy between human and AI capabilities. Keeping up with the pharmaceutical industry which constantly developing and changing will necessitate collaboration and fresh ideas in the field of drug research and development. In the long run, this will lead to better treatments, which will improve people's health all across the globe.

With the use of this technology, it gives more opportunity and availability in regards of medical care in three ways that is, the user-friendliness, modular design, and a complete toolkit. Not forgetting to for our awareness on how DeepChem also have limitations, such as model interpretability and scalability issues shows a commitment to continual improvement. We must also understand that AI and human expertise can

work together to make things better, it ensures the experiments and ongoing development show that the method is dynamic and collaborative. Keeping up with the pharmaceutical industry which constantly developing and changing will necessitate collaboration and fresh ideas in the field of drug research and development. In the long run, this will lead to better treatments, which will improve people's health all across the globe.

To be able to build an environment that is safe and ethical in the field of drug development of artificial intelligence it is highly important we have to foster and create an interdisciplinary collaboration. So with the enforcement of bringing all experts from diverse fields such as computer science, biology, medicine, and ethics will contribute to a practical understanding of the challenges and opportunities. Also, we have to set a goal to be able to prioritize ethical reinforcement in AI development for drug discovery, the guidelines such as ensuring the transparency and fairness of algorithms, addressing bias in datasets, and regularly reassessing how AI can affect the societal impact. Not only enforcing ethical ways will be able to enchance the credibility of AI-driven drug discovery but it will also ensure that advancements benefit society equitably.

The second suggestion we would highlight is, how continuous education and innovation is important in AI drug discoveries. Because the nature of how AI in drug discovery continues to changes and adapt based on the real life situations, there will be a huge importance to be able to commit for continuous innovation. Investment in research and development should be able to focus on refining existing models, overcoming scalability issues, and improving the understanding of AI algorithms.

We would like to also stress how necessary it is to have initiative to promote the educational department to be able to move forward, for instance in situations it is vital to equip researchers, healthcare professionals, and policymakers with the knowledge needed to understand how AI and drug development work around. To be able to do this, we can start by creating a community dedicated to move forward for AI in drug discovery advancement. This community will drive forward and bring enthusiasm to this field by making workshops, training programs, and also promoting the understanding of AI can do in further bringing our society to be a much healthier and safe place to be.

VI. Reference

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.
2. Ali, S., Abuhmed, T., El-Sappagh, S., Muhammad, K., Alonso-Moral, J. M., Confalonieri, R., Guidotti, R., Del Ser, J., Díaz-Rodríguez, N., Herrera, F. (2023). Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence. Information Fusion, 99.
3. Alzubaidi, L., Zhang, J., Humaidi, A. J., et al. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. Journal of Big Data, 8, 53.
4. Cembrowska-Lech, D., Krzemińska, A., Miller, T., Nowakowska, A. Adamski, C., Radaczyńska, M., Mikiciuk, G., Mikiciuk, M. (2023). An Integrated Multi-Omics and Artificial Intelligence Framework for Advanced Plant Phenotyping in Horticulture. Biology, 12(10), 1298.
5. Dara, S., Dhamercherla, S., Jadav, S. S., Babu, C. M., Ahsan, M. J. (2022). Machine Learning in Drug Discovery: A Review. Artificial Intelligence Review, 55(3), 1947–1999.
6. Eastman, P., Ramsundar, B. (2021). Working With Splitters. DeepChem.
7. Feldmann, C., Philipps, M., Bajorath, J. (2021). Explainable machine learning predictions of dual-target compounds reveal characteristic structural features. Scientific Reports, 11, Article 21594.
8. Gerke, S., Minssen, T., Cohen, G. (2020). Ethical and legal challenges of artificial intelligence-driven healthcare. In A. Bohr, K. Memarzadeh (Eds.), Artificial Intelligence in Healthcare (pp. 295–336). Academic Press.
9. Jiménez-Luna, J., Grisoni, F., Schneider, G. (2020). Drug discovery with explainable artificial intelligence. Nature Machine Intelligence, 2, 573–584.
10. Landrum, G. (2010). RDKit.
11. Li, Y., Zhang, L., Wang, Y., Zou, J., Yang, R., Luo, X., Wu, C., Yang, W., Tian, C., Xu, H., Wang, F., Yang, X., Li, L., Yang, S. (2022). Generative deep learning enables

the discovery of a potent and selective RIPK1 inhibitor. Nature Communications, 13, Article 6891.

12. Nag, S., Baidya, A. T. K., Mandal, A., Mathew, A. T., Das, B., Devi, B., Kumar, R. (2022). Deep learning tools for advancing drug discovery and development. 3 Biotech, 12(5), 110.
13. Niazi, S. K., Mariam, Z. (2023). Recent Advances in Machine-Learning-Based Chemoinformatics: A Comprehensive Review. International journal of molecular sciences, 24(14), 11488.
14. Pandey, M., Fernandez, M., Gentile, F., Isayev, O., Tropsha, A., Stern, A. C., Cherkasov, A. (2022). The transformational role of GPU computing and deep learning in drug discovery. Nature Machine Intelligence, 4, 211–221.
15. Pucci, F., Bourgeas, R., Rooman, M. (2016). High-quality Thermodynamic Data on the Stability Changes of Proteins Upon Single-site Mutations. Journal of Physical Chemistry Reference Data, 45, 023104.
16. Ragoza, M., Hochuli, J., Idrobo, E., Sunseri, J., Koes, D. R. (2017). Protein–Ligand Scoring with Convolutional Neural Networks. Journal of Chemical Information and Modeling, 57(4), 942–957.
17. Ramsundar, B. (2021). Molecular Fingerprints. DeepChem.
18. Ramsundar, B. (2021). The Basic Tools of the Deep Life Sciences. DeepChem.
19. Ramsundar, B., Eastman, P., Walters, P., Pande, V., Leswing, K., Wu, Z. (2019). Deep Learning for the Life Sciences. O'Reilly Media.
20. Ramsundar, B., Kearnes, S., Riley, P., Webster, D., Konerding, D., Pande, V. (2015). Massively multitask networks for drug discovery. arXiv preprint arXiv:1502.02072.
21. Reddy, S., Allan, S., Coghlan, S., Cooper, P. (2020). A governance model for the application of AI in health care. Journal of the American Medical Informatics Association: JAMIA, 27(3), 491–497.
22. Selvaraj, C., Chandra, I., Singh, S. K. (2022). Artificial intelligence and machine learning approaches for drug design: challenges and opportunities for the pharmaceutical industries. Molecular Diversity, 26(3), 1893–1913.
23. Singh, A. (2021). DeepChem and its importance in Drug Discovery and Biology. International Journal of Engineering Development and Research (IJEDR), 9(4).
24. Singh, S., Gupta, H., Sharma, P., Sahi, S. (2023). Advances in Artificial intelligence (AI)-assisted approaches in drug screening. Artificial Intelligence Chemistry, 2 (1).

25. Shen, C., Ding, J., Wang, Z., Cao, D., Ding, X., Hou, T. (2019). From machine learning to deep learning: Advances in scoring functions for protein–ligand docking. WIREs Computational Molecular Science.
26. Shiammala, P. N., Duraimutharasan, N. K. B., Vaseeharan, B., Alothaim, A. S., Al-Malki, E. S., Snekaa, B., Safi, S. Z., Singh, S. K., Velmurugan, D., Selvaraj, C. (2023). Exploring the artificial intelligence and machine learning models in the context of drug design difficulties and future potential for the pharmaceutical sectors. Methods, 219, 82-94.
27. Xia, S., Chen, E., Zhang, Y. (2023). Integrated Molecular Modeling and Machine Learning for Drug Design. Journal of Chemical Theory and Computation, 19 (21), 7478-7495.
28. Zeng, X., Wang, F., Luo, Y., Kang, S., Tang, J., Lightstone, F. C., Fang, E. F., Cornell, W., Nussinov, R., Cheng, F. (2022). Deep generative molecular design reshapes drug discovery. Cell Reports Medicine, 3(12).
29. Zhavoronkov, A., Ivanenkov, Y. A., Aliper, A., Veselov, M. S., Aladinskiy, V. A., Aladinskaya, A. V., Terentiev, V. A., Polykovskiy, D. A., Kuznetsov, M. D., Asadulaev, A., Volkov, Y., Zholus, A., Shayakhmetov, R. R., Zhebrak, A., Minaeva, L. I., Zagribelnyy, B. A., Lee, L. H., Soll, R., Madge, D., Xing, L., Guo, T., Aspuru-Guzik, A. (2019). Deep learning enables rapid identification of potent DDR1 kinase inhibitors. Nature Biotechnology, 37(9), 1038–1040.
30. Zinner, M., Dahlhausen, F., Boehme, P., Ehlers, J., Bieske, L., Fehring, L. (2021). Quantum computing's potential for drug discovery: Early stage industry dynamics. Drug Discovery Today, 26(7), 1680-1688.