DeepChem and its importance in Drug Discovery and Biology

1Aditya Singh
1Student
1MIT World Peace University, Pune

Abstract - **There are several different applications of Machine Learning and Deep Learning that can be used for Drug Discovery and Quantum Chemistry. This demonstrates a significant growth of process in medical research and the development of vaccines. In this paper, we will showcase different ways in which we can implement DeepChem for solving problems related to medicines and life sciences.**

keywords - **Deepchem, datasets, training, biotech, deep learning, features, splitters, molecular, graph**

I. INTRODUCTION

DeepChem is used in creating quality tools which are open source and used for drug discovery, materials science, quantum chemistry and biology. We can apply AI in life science because it is a booming industry at present. Using DeepChem will help you in this industry. Diseases are the biggest cause of human suffering. By using the skills helped in building models with the help of DeepChem, we can contribute in the development of new medicine.

Deepchem uses Tensorflow and is an open source framework and is designed to help in the creating of deep learning models for different life sciences applications.

The art of developing new medicines is an elite skills that can be practiced by a small amount of expert practitioners. We will covering the core features of DeepChem, this will help us in the understanding of application of DeepChem for machine learning in healthcare.

II. DATASETS

To work using Machine Learning, datasets are the core requirement for progress. Here we show the dataset which DeepChem uses to store and manage data. It provides simple but powerful tools for efficiently working with large amounts of data.

Design built for easy interaction with other Python frameworks such as Numpy, Pandas, Tensorflow. For the entire working, we are using the Google Colab platform. The different stages of manipulating or creating the dataset are given below.

A. *Dataset Information*

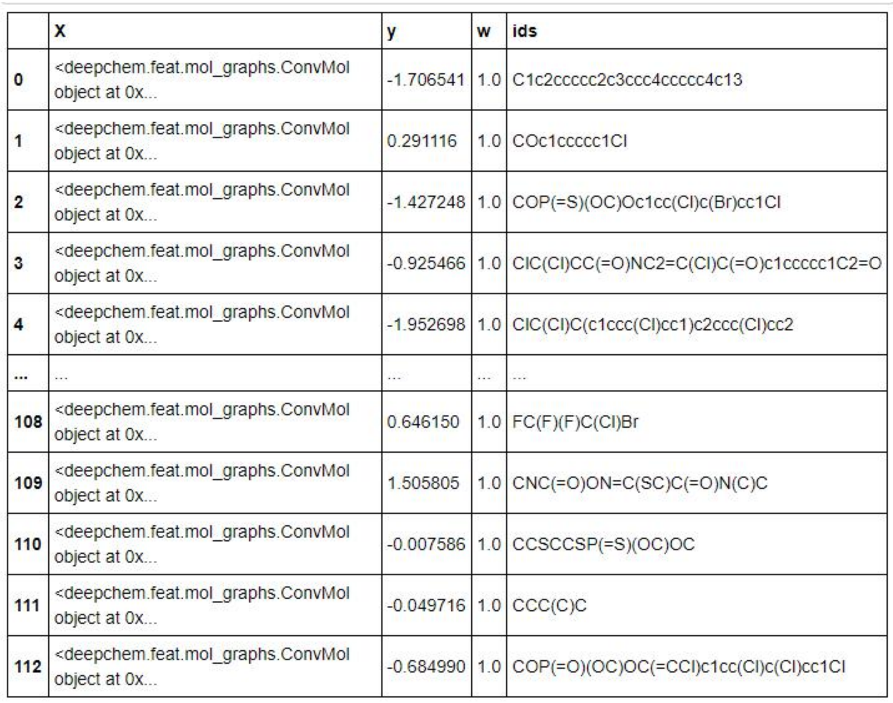
The Disk Dataset is a dataset that has been saved to the disk. The data can be accessed efficiently. Numpy Dataset is the dataset that holds all the data in NumPy arrays. It is helpful when we have to manipulate small to large sized datasets that can contain in the memory. Image Dataset stores all the data in image files on the disk. It is helpful when working with models which have images as their input and output.

B. *Accessing the Data*

The best way to access the dataset is to iterate over it. Load small amounts of data at a time, process the data and then free the memory before adding the next amount. We can make use of the itersamples() method to iterate over samples each time.

Many deep learning models process a batch of multiple samples all at once. We can use iterbatches() to iterate over these batches.

Another way to access the dataset is by using the to_dataframe() function. This copies the data into the DataFrame. This can also be done using small datasets each time.

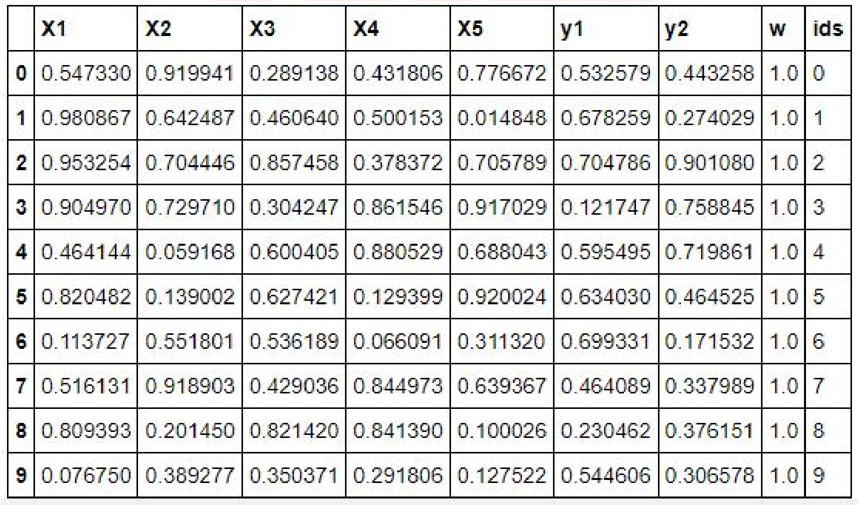


	X	y	w	ids
0	<deepchem.feat.mol_graphs.ConvMol object at 0x...	-1.706541	1.0	C1c2ccccc2c3ccc4ccccc4c13
1	<deepchem.feat.mol_graphs.ConvMol object at 0x...	0.291116	1.0	COc1ccccc1Cl
2	<deepchem.feat.mol_graphs.ConvMol object at 0x...	-1.427248	1.0	COP(=S)(OC)Oc1cc(Cl)c(Br)cc1Cl
3	<deepchem.feat.mol_graphs.ConvMol 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(=CCl)c1cc(Cl)c(Cl)cc1Cl

Figure1. Final Data Frame

C. *Creating a Dataset*

We can create our own dataset using NumpyDataset. We can start passing the arrays containing the data and then wrapping them up in a NumpyDataset. The output of the data frame is given below.



	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	0.918903	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

Figure2. Data Frame after Creation

III. METHODOLOGY

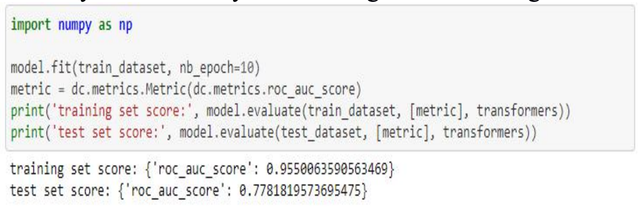
A. *Molecular Fingerprints*

Molecules are represented in different ways. This method shows us a type of representation called as "Molecular Fingerprint". It is a representation that works for small drug like molecules.

DeepChem takes a particular type of fingerprint called an "Extended Connectivity Fingerprint" or ECFP. This ECFP algorithm starts by classifying atoms based on their properties and bonds that occurs. Each pattern in is a particular feature. For eg:- Carbon atom bonded to two hydrogen and two heavy atoms would be a feature.

We start by training a model. Hence, we start off by using a fixed length array which is a simpler type of model.

A MultiTaskClassifier is a stack of fully connected layers. Training and evaluating of the model is given below.



```
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': 0.9550063590563469}
test set score: {'roc_auc_score': 0.7781819573695475}
```

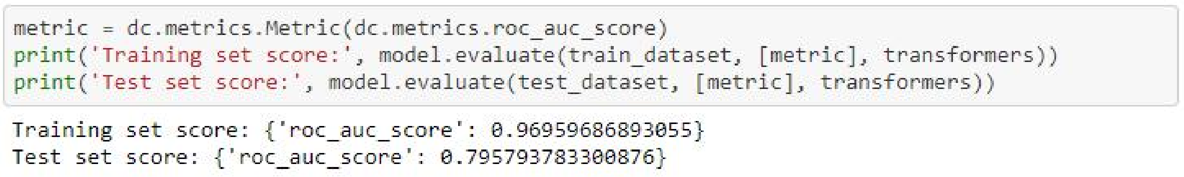
Figure3. Results of the model

B. *Graph Convolutions*

Here, we are showing the significance and working of Graph Convolutions. This is the most powerful deep learning tools for molecular data. We can view molecules with respect to graphs. It is a normal CNN (Convolutional Neural Network) for processing images using grid of pixels. Every layer combines the data from a pixel to produce a new data vector for the pixel. The starting layers detect small patterns, while end layers detect larger, more abstract patterns. Often the convolutional layers alternate with pooling layers that perform some operation such as max or min over local regions.

To train the GraphConvModel we use the MoleculeNet for loading the dataset.

After training the dataset we evaluate the performance of the model. We need to define a metric, measure used to represent model performance. We will use the ROC-AUC score. Moreover, we use the model.evaluate() function to check our score and accuracy.



```
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))
```

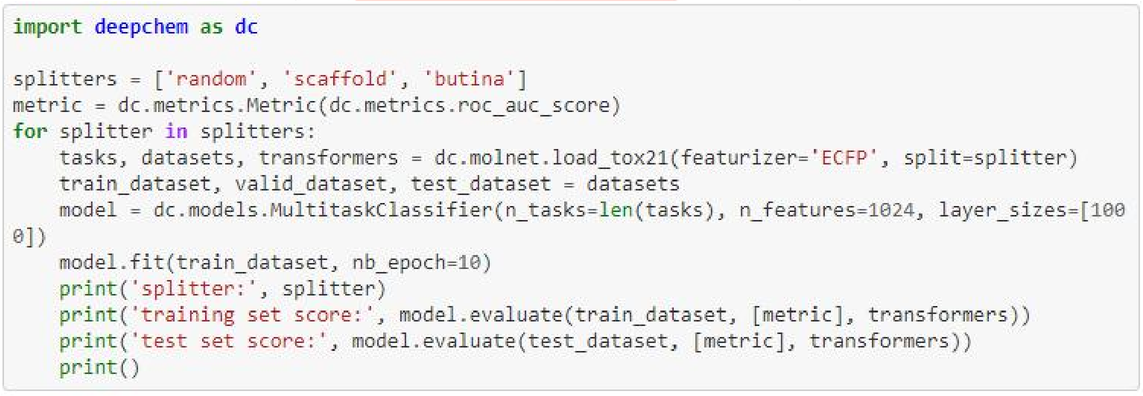
```
Training set score: {'roc_auc_score': 0.96959686893055}
Test set score: {'roc_auc_score': 0.795793783300876}
```

Figure4. Results of the model

C. *Splitting up datasets*

Generally, when we use machine learning tools on datasets, we divide it into training, testing test data sets. We can do this automatically using the MoleculeNet. We will be showcasing some splitting methods that can be used for splitting up datasets using DeepChem.

In DeepChem, we use the 'Splitter' object to split samples into multiple datasets. We are taking a dataset to test the different splitters that can be implemented. We will be using the Tox21 dataset.



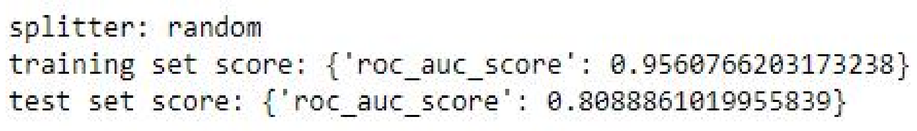
```
import deepchem as dc

splitters = ['random', 'scaffold', 'butina']
metric = dc.metrics.Metric(dc.metrics.roc_auc_score)
for splitter in splitters:
    tasks, datasets, transformers = dc.molnet.load_tox21(featurizer='ECFP', split=splitter)
    train_dataset, valid_dataset, test_dataset = datasets
    model = dc.models.MultitaskClassifier(n_tasks=len(tasks), n_features=1024, layer_sizes=[100
0])
    model.fit(train_dataset, nb_epoch=10)
    print('splitter:', splitter)
    print('training set score:', model.evaluate(train_dataset, [metric], transformers))
    print('test set score:', model.evaluate(test_dataset, [metric], transformers))
    print()
```

Figure5. Spliiters that we are implementing

The different types of splitters offered by DeepChem (currently using) are as follows: -

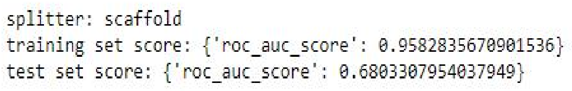
- **Random Splitter: -** Selecting samples for training, validation and testing sample in a random manner
.



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

Figure6. Random Splitter performance

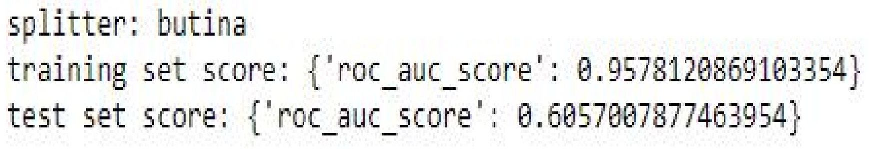
- **Scaffold Splitter: -** Sometimes, molecules can be similar to each other, hence, ScaffoldSplitter finds out the scaffold that creates at the center of each molecule and makes sure that all molecules with the same scaffold are inserted into the same data set.



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

Figure7. Scaffold Splitter performance

- **ButinaSplitter: -** ButinaSplitter combines or clusters molecules based on their fingerprints. Hence, the ones with same fingerprints will be in the same dataset.



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

Figure8. Butina Splitter performance

IV. Conclusion

In this research work, we have seen a small fraction of applications with respect to DeepChem and its scope. The different ways in which DeepChem is creating high performance computations using deep learning algorithms in the fields of drug discovery, material science and biology has been impeccable.

We have seen different ways in which we can use DeepChem for machine learning in healthcare. From different data manipulation and creation techniques to molecular fingerprints, we have taken an in depth guide into the actual working of DeepChem and its abundant applications.

DeepChem can be evolved into a powerful framework for science that is based on AI and ML. The current design or workflow starts from inputs and the outputs and performing a sequence of operations. For example, Dataset creation can be routed to a model filtered by a Splitter or a Transformer in between this process.

Hence, DeepChem gives machine learning a domain specific language. Meaning, depending on what type of splitter, loader, transformer, model, hyper parameter we choose, we have a different program based on our choice.

However, DeepChem has some limitations. DeepChem is linear, which means imput flows through single series transformations to produce outputs. DeepChem continues to evolve in making flexible domain specific language for machine learning methods in creating new applications.

V. Future Work

In the future, we will be implementing and researching on DeepChem related functionalities and applications even more.

We will be covering topics like: -

- Molecular Featurizations
- Unsupervised Embedding for Molecules
- Atomic Contributions for Molecules
- Large Scale Chemical Screens
- Protein Deep Learning
- Quantum Chemistry
- Bioinformatics
- Reinforcement Learning
- Burgers Equation for Physics informed Neural Networks

References

1. K. Shailaja, B. Seetharamulu and M. A. Jabbar, "Machine Learning in Healthcare: A Review," *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 2018, pp. 910-914, doi: 10.1109/ICECA.2018.8474918.
2. Shailaja, K. & Seetharamulu, B. & Jabbar, M.. (2018). Machine Learning in Healthcare: A Review. 910-914. 10.1109/ICECA.2018.8474918.
3. Sidey-Gibbons, J., Sidey-Gibbons, C. Machine learning in medicine: a practical introduction. *BMC Med Res Methodol* **19,** 64 (2019). https://doi.org/10.1186/s12874-019-0681-4\
4. Saini, Akanksha and Meitei, A J and Singh, Jitenkumar, Machine Learning in Healthcare: A Review (April 26, 2021). Proceedings of the International Conference on Innovative Computing & Communication (ICICC) 2021, Available at SSRN: https://ssrn.com/abstract=3834096 or http://dx.doi.org/10.2139/ssrn.3834096
5. Miotto R, Wang F, Wang S, Jiang X, Dudley JT. Deep learning for healthcare: review, opportunities and challenges. *Brief Bioinform.* 2018;19(6):1236-1246. doi:10.1093/bib/bbx044
6. Tobore I, Li J, Yuhang L, Al-Handarish Y, Kandwal A, Nie Z, Wang L Deep Learning Intervention for Health Care Challenges: Some Biomedical Domain Considerations JMIR Mhealth Uhealth 2019;7(8):e11966 URL: https://mhealth.jmir.org/2019/8/e11966DOI: 10.2196/11966
7. Esteva, A., Chou, K., Yeung, S. *et al.* Deep learning-enabled medical computer vision. *npj Digit. Med.* **4,** 5 (2021). https://doi.org/10.1038/s41746-020-00376-2

8. Z. Zheng, Y. Liu, Y. Zhang and C. Wen, "TCMKG: A Deep Learning Based Traditional Chinese Medicine Knowledge Graph Platform," *2020 IEEE International Conference on Knowledge Graph (ICKG)*, 2020, pp. 560-564, doi: 10.1109/ICBK50248.2020.00084.
9. S. Zhang, S. M. H. Bamakan, Q. Qu and S. Li, "Learning for Personalized Medicine: A Comprehensive Review From a Deep Learning Perspective," in IEEE Reviews in Biomedical Engineering, vol. 12, pp. 194-208, 2019, doi: 10.1109/RBME.2018.2864254.
10. S. Roy et al., "Deep Learning for Classification and Localization of COVID-19 Markers in Point-of-Care Lung Ultrasound," in IEEE Transactions on Medical Imaging, vol. 39, no. 8, pp. 2676-2687, Aug. 2020, doi: 10.1109/TMI.2020.2994459.
11. S. Zhang, S. M. H. Bamakan, Q. Qu and S. Li, "Learning for Personalized Medicine: A Comprehensive Review From a Deep Learning Perspective," in IEEE Reviews in Biomedical Engineering, vol. 12, pp. 194-208, 2019, doi: 10.1109/RBME.2018.2864254.
12. W. Chang et al., "A Deep Learning Based Wearable Medicines Recognition System for Visually Impaired People," 2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS), 2019, pp. 207-208, doi: 10.1109/AICAS.2019.8771559.
13. S. Hussein, P. Kandel, C. W. Bolan, M. B. Wallace and U. Bagci, "Lung and Pancreatic Tumor Characterization in the Deep Learning Era: Novel Supervised and Unsupervised Learning Approaches," in *IEEE Transactions on Medical Imaging*, vol. 38, no. 8, pp. 1777-1787, Aug. 2019, doi: 10.1109/TMI.2019.2894349.