MiNNiTox: a deep learning pipeline for predicting the toxicity of novel molecules

Nikolaos Gialitsis
DIT, NKUA
BRFAA

Abstract

We examine the effect of neural network architectures in the task of classifying molecules as toxic or non-toxic. We formulate the task of toxicity prediction as a binary classification problem, and we test a variety of deep learning architectures, exploring a range of different settings for classification and modelling. A preliminary investigation via a wide experimental evaluation on the "Smiles Toxicity" dataset from Kaggle illustrates that colored-image representations of the molecules can prove beneficial to the classification task, compared to a typical Quantitative Structural Activity Relationship (QSAR) type of representation, which focuses on the structural similarity between the molecules, with Convolutional Neural Networks providing the best results for AUC in the ROC curve [work in progress].

1 Introduction

[TEMPORARY] The phrase "Deep learning" has gained buzzword status in recent years. Regardless of whether the unprecedented hype surrounding technique is well placed, it has established itself as state of the art in many fields, most notably in image and speech recognition and natural language processing, where tech giants have invested billions of dollars in driving the technology forward.

The development of a consistent and time-efficient method of toxicity prediction can assist pharmaceutical companies and biomedical scientists in their day to day tasks, as well as provide insight into the molecular mechanisms and interactions that determine the effects of drugs, cosmetics, and pollutants.

2 Related work

2.1 Deep Learning

[TEMPORARY]
Deep Learning is known to learn abstract representations of the input data with higher levels of abstractions in higher layers (LeCun et al., 2015). This concept has been relatively straightforward to demonstrate in image recognition, where simple objects, such as edges and simple blobs, in lower layers are combined to abstract objects in higher layers (Lee et al., 2009). In toxicology, however, it was not known how the data representations from Deep Learning could be interpreted.

Similar to the successes in other fields (Dahl et al., 2012; Krizhevsky et al., 2012; Deng et al., 2013; Graves et al., 2013; Socher and Manning, 2013; Baldi et al., 2014; Sutskever et al., 2014), Deep Learning has increased the predictive performance of computational methods in toxicology. As confirmed by the NIH1, the high quality of the models in the Tox21 challenge makes them suitable for deployment in leading-edge toxicological research. We believe that Deep Learning is highly suited to predicting toxicity and is capable of significantly influencing this field in the future.

2.2 Quantitative Structure Activity Relationship (QSAR)

[TEMPORARY]
In QSAR modeling, the predictors consist of physico-chemical properties or theoretical molecular descriptors of chemicals; the QSAR response-variable could be a biological activity of the chemicals. QSAR models first summarize a supposed relationship between chemical structures and biological activity in a data-set of chemicals. Second,

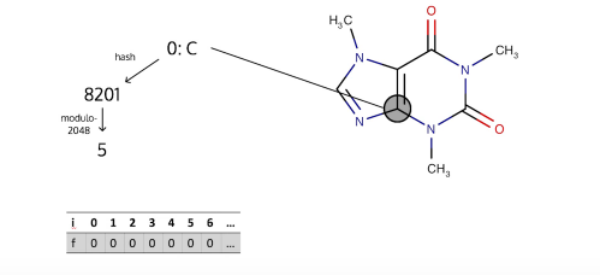

Figure 1: 1st step of morgan fingerprint: drawing a circle of radius R centered around a carbon atom, and then hashing all molecules residing within the circle allows the representation of the molecule as a numerical vector. Each time an atom is hashed into a cell, the cell's number is incremented by one

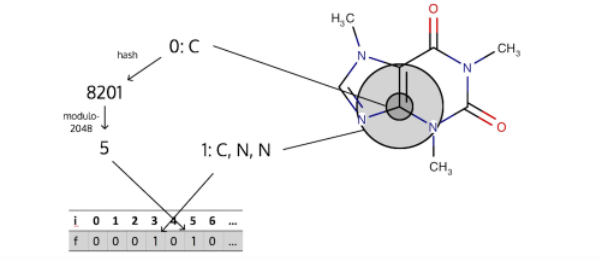

Figure 2: 2nd step of morgan fingerprint: we repeat the application of the hash-function on molecules within a circle with double the radius aka. 2*R

QSAR models predict the activities of new chemicals

2.3 Tox21 Data Challenge

Deep Learning has been shown to be state of the art for QSAR problems, including winning the Kaggle Merck Molecular Activity Challenge, and the recent Tox21 toxicity challenge.

The challenge: The latter consisted of a dataset containing 12K environmental chemicals and drugs, and each is considered a candidate for causing one ore more of 12 different toxic effects. These effects are characterized into belonging to two major groups, the stress response effects (SR) and the nuclear receptor effects (NR). These in union, include factors such as disruption of estrogen receptors or malfunctions in the p53 pathway which is linked to tumor-suppression. Serious medical conditions such as liver injury or cancer and disruption of the function of the endocrinic

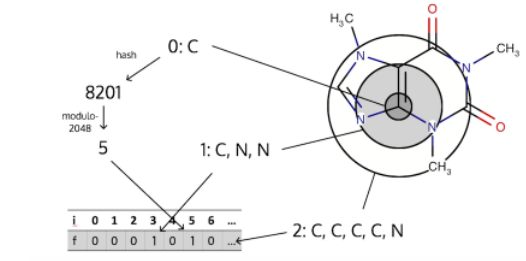

Figure 3: 3d step of morgan fingerprint: the circle is now 3*R and most atoms are hashed. The process continues for n steps

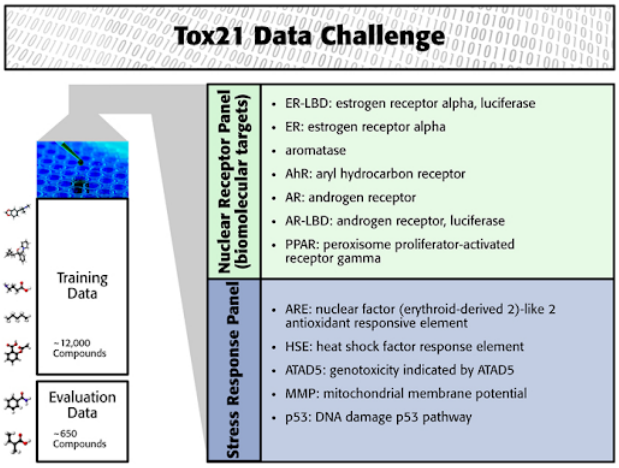

Figure 4: the toxic effects for which the compounds were tested on

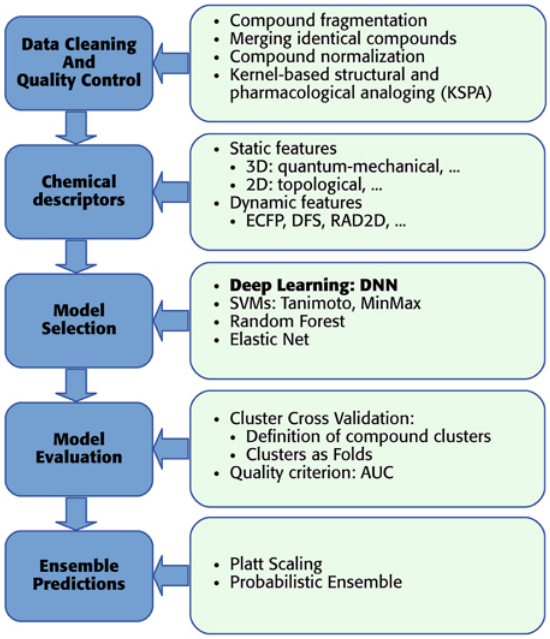

Figure 5: the deeptox pipeline

system can be the result of the NR, and SR effects respectively.

[need to rephrase]

Multi-task learning The construction of indicative abstract features by Deep Learning can be improved by Multi-task learning. Multi-task learning incorporates multiple tasks into the learning process (Caruana, 1997). In the case of DNNs, different related tasks share features, which therefore capture more general chemical characteristics. In particular, multi-task learning is beneficial for a task with a small or imbalanced training set, which is common in computational toxicity. In this case, due to insufficient information in the training data, useful features cannot be constructed. However, multi-task learning allows this task to borrow features from related tasks and, thereby, considerably increases the performance.

DeepTox pipeline The steps that they followed are depicted in figure 2.3.

2.4 Vector Space Models

Vector Space Model (VSM) approaches project the input to a n-dimensional vector representation, where the semantic similarity of the elements is determined by their distance (e.g cosine, euclidean, etc.) in the projected vector space. Feature vector representations are widely used in Machine Learning tasks, e.g. for classification, clustering, etc. of a collection of input items. In the case of toxicity prediction, the molecules are projected into a ”chemical” numerical space which reflects their similarity.

3 Proposed Method

[need to cite sources related to CNNs]

3.1 Molecules as colored images

In the QSAR-based predictions that were discussed in the previous section, the substances are represented with molecular fingerprints, which are based on the 2D conformation of the atoms. However, as the molecules are naturally 3-Dimensional, this means that the spatial space is simplified, and important information is lost in the process.

For this reason, the approach I followed is to embed the molecules in three-dimensional space, and then represent them in the RGB system as colored pixels (Red,Green,Blue). Convolutional Neural Networks have been very successful in image classification, and the intuition is that by representing each molecule as a colored-image, a CNN can be used for the task of toxicity prediction. Another important reason for which CNNs are preferred, is that they allow the representations to be independent of the exact positions of the pixels aka. the atoms in the molecule. This attribute is why a handwritten digit classifier can recognize a number, independently of it’s angle or exact position in the image. This is the result of the sliding window filters and pooling that takes place in a CNN which allow the models to identify high-level features from the input.

3.2 Retrieving molecular information

Dataset The ”SMILES” Toxicity dataset from Kaggle contains toxicity information of 7960 molecules which are further split into 7696 in the training set and 264 in the testing set. A label of zero means that the molecule is non-toxic and a label of one indicates toxicity. The molecules are represented with SMILES strings. The simplified molecular-input line-entry system (SMILES) describes the structure of chemical entities using short ASCII strings.

3.3 Data Preprocessing and Cleansing

Before continuing further with the representation and analysis of the data, the tables provided in the dataset needed to be configured to fit the classification paradigm. More specifically, the desired

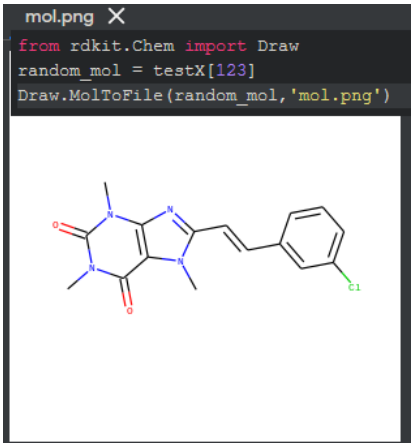

Figure 6: visualization of molecule in testing set

```
MODEL      109
COMPND     UNNAMED
AUTHOR     GENERATED BY OPEN BABEL 2.4.1
HETATM    1 NA   UNL     1       2.368   1.336  -3.395  1.00  0.00          Na1+
HETATM    2 NA   UNL     2       1.798  -0.246   3.651  1.00  0.00          Na1+
HETATM    3  C   UNK A   0       3.109  -3.296   2.727  1.00  0.00           C
HETATM    4  N   UNK A   0       1.965  -2.394   2.442  1.00  0.00           N
HETATM    5  C   UNK A   0       1.372  -2.587   1.046  1.00  0.00           C
HETATM    6  C   UNK A   0       2.404  -2.568  -0.135  1.00  0.00           C
HETATM    7  N   UNK A   0       3.151  -3.804  -0.293  1.00  0.00           N
HETATM    8  C   UNK A   0       2.509  -5.014  -0.569  1.00  0.00           C
HETATM    9  C   UNK A   0       1.143  -5.095  -0.866  1.00  0.00           C
HETATM   10  C   UNK A   0       0.473  -6.310  -0.841  1.00  0.00           C
HETATM   11  C   UNK A   0       1.162  -7.491  -0.568  1.00  0.00           C
HETATM   12  C   UNK A   0       2.548  -7.449  -0.393  1.00  0.00           C
HETATM   13  C   UNK A   0       3.212  -6.213  -0.381  1.00  0.00           C
HETATM   14  C   UNK A   0       0.347  -8.725  -0.468  1.00  0.00           C
HETATM   15  O   UNK A   0      -0.811  -8.700  -0.884  1.00  0.00           O
ATOM     16  N   GLU A   1       0.937  -9.779   0.160  1.00  0.00           N
ATOM     17  CA  GLU A   1       0.256 -11.060   0.446  1.00  0.00           C
ATOM     18  CB  GLU A   1       0.573 -12.088  -0.654  1.00  0.00           C
ATOM     19  CG  GLU A   1       0.148 -11.618  -2.052  1.00  0.00           C
ATOM     20  CD  GLU A   1       0.499 -12.500  -3.245  1.00  0.00           C
ATOM     21  OE1 GLU A   1       1.069 -13.593  -3.006  1.00  0.00           O1-
```

Figure 7: snippet of the pdb file produced by OpenBabel

table requires two columns, the SMILES and the Labels. This was not supplied a priori so it had to be done manually by merging the tables.

Furthermore, entries containing ”None” fields were ignored. Then, the molecules were transformed into Molecular Fingerprints using the Morgan Fingerprint method from the rdkit.Chem python library. Some molecules were in an invalid form, and thus the transformation failed so those molecules were also excluded. Using this chemoinformatics toolkit, I transformed the SMILES into python classes. These chemical classes support a range of utilities such as the ability to visualize the molecule (Figure 3.3).

Thus the remaining molecules were represented as 2048 bit vectors, in a similar fashion to the Vector Space Models discussed in the Related Work segment.

3.4 Extracting 3D-coordinates

The Rdkit library has a function for retrieving the atom coordinates from a molecule, but it seems that the third-dimension is always zero for all molecules. I assume this is because the initial data from kaggle was in SMILE strings which are only 2D-representations.

Thus, I ended up using a different tool called OpenBabel for this puprose. After saving all the molecular classes in a single file, I loaded the file into OpenBabel with the suitable command argument corresponding to 3D generation.

The output was a protein data bank file format which contains the information portrayed in figure 3.4. I used the Jena3D pdb viewer [1] to visualize

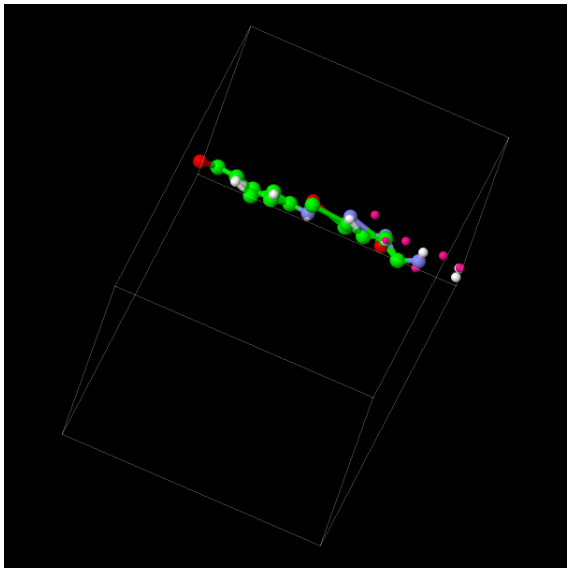

Figure 8: PDB representation

the conformation of a random molecule (Figure 3.4.

The xyz spatial coordinates correspond to the last three fields in every line of the file. Thus, these had to be manually retrieved with some string manipulation techniques (depicted in Figure 6).

[Work in Progress]

3.5 Mapping coordinates to colors

[1] jena3d.leibniz-fli.de/

4 Experiments

[Work in progress]

4.1 Over-sampling

Since the dataset used contains very unbalanced classes – the grand majority (with a ratio approximately 7 to 1) belonging to class 0, aka. the non-toxic molecules. To alleviate this, we employ an oversampling scheme using SMOTE to arrive at a 2-1 ratio at most. [2].

4.1.1 Partitioning the dataset into training and testing sets

Any supervised machine learning procedure involves training a model on an input training set and then predicting the values on an unseen testing set. There exist two popular approaches we could use: the Holdout method or the Cross-validation method. In the first approach, one percentage of the data is simply selected as a the training set and the rest consists the testing set(e.g 90% training and 10% testing). However such an approach could introduce bias into the experiments, because there can be a significant difference between the results obtained for two different partitions of the same dataset. The k-fold cross-validation method combats this issue by splitting the dataset into k equal sized partitions, and executing the algorithm k times- each time selecting one different partition as the testing set- and keeping the rest k-1 partitions as the training set. This method tries to balance the bias caused by the randomness on the selection of the testing set and allows averaging the results obtained by each run- thus reducing the effect of outliers. For this reason, 10-fold cross-validation was performed when testing our classifiers on the Morgan Fingerprints and 3D representations.

[Work in progress]

4.2 Evaluation

The evaluation is performed in terms of AUC; the former is calculated by counting the area below the ROC curve.

[Work in Progress]

[2] imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTE.html

5 Results and Discussion

Statistical testing in R

Shapiro-Wilk Normality Check The Shapiro-Wilk normality test was applied on the classification results for each experiment, and the p-values are recorded.

Parametric vs Non-parametric test if p-value < 0.05 in the normality test, proceed with ANOVA otherwise proceed with Kruskal-Wallis

Posthoc analysis

6 Conclusions

```
Xtrain_coords = []
model_num = 1
array = []
with open('molecule.pdb') as pdbfile:
    for line in pdbfile:
        if line[:5] == 'MODEL':
          print('Parsing  Molecule '+str(model_num))
          model_num = model_num + 1
          array = []
        elif line[:4] == 'ATOM' or line[:6] == "HETATM":
            #print(line)
            # Split the line
            x_list = line[30:38]
            y_list = line[38:46]
            z_list = line[46:54]

            x_str = ' '.join([str(elem) for elem in x_list])
            y_str = ' '.join([str(elem) for elem in y_list])
            z_str = ' '.join([str(elem) for elem in z_list])

            x_str_no_space = x_str.replace(" ", "")
            y_str_no_space = y_str.replace(" ", "")
            z_str_no_space = z_str.replace(" ", "")

            x_float = float(x_str_no_space)
            y_float = float(y_str_no_space)
            z_float = float(z_str_no_space)

            #print('x = '+str(x_float),'y = '+str(y_float), 'z = ' + str(z_float))
```

Figure 9: code for 3D-coordinate extraction

```
import tensorflow.keras as keras
from keras import Model
from keras.layers import Activation, Dense, Dropout, Input
from keras.utils import np_utils
from rdkit.Chem import DataStructs

i = 0
trainX=[]
ignore = [334,5223,6374,6880] #rdkit returns none
for mol in train_smiles :
  if mol is not None:
    if i not in ignore:
      m = Chem.MolFromSmiles(mol)
      assert(m is not None)
      finger = AllChem.GetHashedMorganFingerprint(m,2,nBits=2048)
      array = np.zeros((0, ), dtype=np.int8)
      DataStructs.ConvertToNumpyArray(finger, array)
      trainX.append(array)
    i = i+1

print(str(len(trainX))+" sanitized molecules in the training set")
```

Figure 10: python code depicting data cleansing and representation