

Summer 2018 Project Report

# **Predicting Binding Affinities of Proteins and Ligands with Deep Learning Architectures**

---

Student Name: Sachita Nishal

Under Supervision of: Dr. Karthik Raman

Project Duration: May 2018 - July 2018



## Introduction

Structure-based screening methods in drug design work by predicting a binding affinity between a target and candidate molecule, based on the 3D complex structure. This has allowed researchers to rank candidates, and hence prioritize them accordingly, for laboratory testing for drug design. Recently, however, there has been a shift towards prediction of binding affinities using Machine Learning techniques.

However, even specifically in machine learning, the application of Deep Learning has shown considerable promise for this problem. Deep learning helps to deal easily with the large amounts of structural data that are increasingly becoming publicly available. Deep Learning also reduces the need for feature engineering (which is generally a requirement for Machine Learning methods). Instead, the model learns to ‘weigh’ features as a natural consequence of the process of fitting the model’s parameters to the available data. This weighting of features is what affects decision-making during predictions. Hence, choosing the representation of the input data has a major impact on the predictive power of a model. Additionally, Deep Neural Networks allow to easily create multitask classifiers or regressors, that can predict, for instance, activities against multiple targets at once.

Over the summer, I explored various papers that used different feature selection methods and Convolutional Neural Network architectures in order to carry out protein-ligand scoring. This report recounts literature reviews of some of these papers I read, and then seeks to explore in-depth a particular paper that I reviewed and worked to replicate. This is followed by a discussion of the future scope of the work carried out on this project.

## Literature Reviews for Feature Selection

(excerpts from papers reviewed, and slides presented on 6 June)

### 1. *AtomNet: A Deep Convolutional Neural Network for Bioactivity Prediction in Structure-based Drug Discovery*<sup>1</sup> (2015)

- a. The first **structure-based, deep convolutional neural network** designed to predicting molecular binding affinity for drug discovery applications
- b. Modelled the prediction problem as a **classification task** where it outputted whether a given protein-ligand structure would bind or not
- c. **Features** incorporated into model:
  - i. **input layer receives vectorised versions of 1Å° 3D grids** placed over co-complexes of the target proteins and small-molecules that are **sampled within the target's binding site** (complex undergoes several **transformations** like shifting coordinates, centring, cropping etc. Multiple poses are also sampled within the binding site cavity.)
  - ii. Each grid cell holds a value that **represents the presence of some basic structural features in the location**. The grid is then unfolded to a 1D vector.
  - iii. Basic structural features can vary from a simple enumeration of atom types to more complex protein-ligand descriptors such as SPLIF, SIFt, or APIF. In depth explanations of these are given below.
  - iv. SIFt: Structural Interaction Fingerprint <sup>2</sup>
    1. **Molecular docking results** are selected (**with multiple poses**) and they **identify a list of binding site residues that are common in all complex structures being studied**. The ligand binding site is defined as the **union of all of the residues that are in contact with**

<sup>1</sup> "AtomNet: A Deep Convolutional Neural Network for Bioactivity ...." 10 Oct. 2015, <https://arxiv.org/abs/1510.02855>. Accessed 14 Aug. 2018.

<sup>2</sup> "Structural interaction fingerprint (SIFt): a novel method for ... - NCBI." <https://www.ncbi.nlm.nih.gov/pubmed/14711306>. Accessed 14 Aug. 2018.

any ligand molecules in any of the structures in the group.

2. After all of the ligand binding site residues were identified and **all of the protein-ligand intermolecular interactions were calculated**, the next step was to **classify these interactions**. **Seven different types of interactions** occurring at each binding residue were extracted and classified:

(i) whether it is in contact with the ligand; (ii) whether any main chain atom is involved in the contact; (iii) whether any side chain atom is involved in the binding; (iv) whether a polar interaction is involved; (v) whether a nonpolar interaction is involved; (vi) whether the residue provides hydrogen bond acceptor(s); and (vii) whether it provides hydrogen bond donor(s).

3. By doing so, **each residue was represented by a seven bit long bit string**. Therefore, interaction fingerprints are of the same length and each bit in the fingerprint represents the presence or absence of a particular interaction at a particular binding site.

v. SPLIF: Structural Protein Ligand Interaction Fingerprints<sup>3</sup>

1. In the first step, **a reference protein-bound ligand is inspected for protein–ligand contacts**. Two atoms are considered being in a contact if the **distance between them is within a specified threshold** (4.5 Å in this study).
2. In the second step, for each ligand– protein atom pair, **the respective ligand and protein atoms are expanded to circular fragments**, i.e., fragments that **include the atoms in question and their successive neighbourhoods up to a certain distance**
3. Each type of circular fragment is **assigned an identifier**. Here, we made use of **Extended Connectivity Fingerprints up to the first closest neighbor (ECFP2)**. It simply **retains information about all atom/ bond types** and **uses one unique integer identifier to**

---

<sup>3</sup> "Structural Protein–Ligand Interaction Fingerprints - ACS Publications." Accessed August 14, 2018. <https://pubs.acs.org/doi/pdf/10.1021/ci500319f>.

**represent one substructure** (i.e., circular fragment). For each atom, the following properties are considered by default:

- a. the atomic number;
- b. the number of "heavy" (non-hydrogen) neighbor atoms;
- c. the number of attached hydrogens (both implicit and explicit);
- d. the formal charge;
- e. an additional property that indicates whether the atom is part of at least one ring.

4. In the third step, **3D coordinates are retrieved for all atoms involved in ligand and protein fragments.**
5. The major difference of SPLIF from SIFt is that **in SPLIF the interactions are encoded implicitly, as a result of explicitly encoding ligand and protein fragments**, whereas in SIFt-like methods the interaction types need to be encoded explicitly, by means of empirical rules. Consequently, **most of current SIFt-like implementations handle only a small number of interaction types.**

vi. APIF : Atom Pairs Based Interaction Fingerprint<sup>4</sup>

1. A new interaction fingerprint (IF) called APIF has been developed for post-processing protein–ligand docking results.
2. Unlike other existing fingerprints which employ absolute locations of individual interactions, **APIF considers the relative positions of pairs of interacting atoms.**

d. **Results:**

- i. The locally-constrained deep convolutional architecture allows the system to model the complex, non-linear phenomenon of molecular binding by **hierarchically composing proximate basic chemical features into more intricate ones.**

---

<sup>4</sup> "APIF: A New Interaction Fingerprint Based on Atom ... - ACS Publications." Accessed August 14, 2018. <https://pubs.acs.org/doi/abs/10.1021/ci900043r>.

- ii. By incorporating structural target information, AtomNet can **predict new active molecules even for targets with no previously known modulators.**
- iii. The paper uses the **area under the receiver operating characteristic (AUC)** and logAUC to report results over the three benchmarks. The AUC indicates classification (or ranked order) performance by measuring the **area under the curve of true-positive rate versus the false-positive rate.** AUC value of 1.0 means perfect separation whereas a value of 0.5 implies random separation.
- iv. AtomNet shows outstanding results on a widely used structure-based benchmark achieving an **AUC greater than 0.9 on 57.8% of the targets in the DUDE benchmark<sup>5</sup>**, far surpassing previous docking methods (compared with Surflex Dock, Dock3.7, Dock6.7, all referenced in the paper).

## 2. CGBVS-DNN: Prediction of Compound-protein Interactions Based on Deep Learning<sup>6</sup> (2017)


- a. CGBVS stands for Chemical Genomics Based Virtual Screening<sup>7</sup>, which was originally based on a Support Vector Machine
- b. The main advantage of using CGBVS is that it **enables CPIs (Compound-Protein Interactions) of novel ligands without known three-dimensional structures to be predicted.**
- c. CGBVS shows high performance in the cross-validation of over 250 thousand datasets containing 125 thousand positive interaction datasets the researchers collected and 125 thousand negative datasets they artificially generated.

---

<sup>5</sup> "DUD-E: A Database of Useful (Docking) Decoys — Enhanced." Accessed August 14, 2018. <http://dude.docking.org/>.

<sup>6</sup> "CGBVS-DNN: Prediction of Compound-protein Interactions ... - NCBI." Accessed August 14, 2018. <https://www.ncbi.nlm.nih.gov/pubmed/27515489>.

<sup>7</sup> "Analysis of multiple compound–protein interactions reveals novel ...." Accessed August 14, 2018. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3094066/>.

- 
- d. However, the size of available datasets can be in the millions, and since SVMs require an exponential increase in resources to cope, modelling with Deep Neural Nets has been used to advance the research.
- e. Features and Data:
- i. Collected more than 2 million pieces of positive interaction data between compounds and proteins in the family of G-protein coupled receptors (GPCR).
  - ii. Compound-protein interactions (CPIs) represent a credible data set because **only interactions with relatively high affinities were selected (K<sub>i</sub>, EC<sub>50</sub>, and IC<sub>50</sub> < 30 μM).**
  - iii. **Artificially generated negative data** by changing combinations between the two pairs of proteins and compounds picked at random from the positive dataset (process detailed in diagram on Page 8)
  - iv. **Molecular and protein descriptors were further obtained** from this bioactivity data, in conjunction with the sequence descriptors. We can focus on how the molecular descriptors are obtained and what features they encapsulate, since sequence data is not our primary interest in this project. The DRAGON software is used for this<sup>8</sup>.
  - v. In DRAGON molecule representation, **atoms are characterised by their atomic number, connectivity (i.e. the number of bonded atoms), valence, charge (if specified in the input file), and some atomic properties encoding chemical information.**
  - vi. The main atomic parameters used as the atomic weightings for molecular descriptor calculations are mass, van der Waals volume, Sanderson electronegativity and polarisability. All the weights are scaled with respect to the carbon atom.
  - vii. The paper uses an **894 dimensional chemical descriptor feature vector** and although it is unspecified which ones of **DRAGON's 1600 possible**

---

<sup>8</sup> "DRAGON Software: An Easy Approach to Molecular Descriptor." Accessed August 14, 2018. [http://match.pmf.kg.ac.rs/electronic\\_versions/Match56/n2/match56n2\\_237-248.pdf](http://match.pmf.kg.ac.rs/electronic_versions/Match56/n2/match56n2_237-248.pdf).

**molecular descriptors** are used to generate this, the possibilities are endless:

1. **Constitutional descriptors**: no. of atoms, bonds, rings etc.
  2. **Enumerative descriptors**: counts of functional groups
  3. Topological and Topographic descriptors:
    - a. **Topological descriptors** are based on **graph representation of molecule**. Numerical quantifiers of molecular topology obtained by the application of algebraic operators to matrices representing molecular graphs and whose values are independent of vertex numbering or labelling. They can be **sensitive to one or more structural features of the molecule such as size, shape, symmetry, branching and cyclicity** and can also encode chemical information concerning atom type and bond multiplicity.
    - b. **Topographic indices** are **derived from the graph representation of molecules** in the same way as the topological indices, but **using the geometric distances between atoms instead of the topological distances**.
  4. Some more descriptors based on **3D structure** and **literature models** (eg. Murigochi logP, Lipinski rule-of-five etc.)
- viii. Process for generating negative data, as represent pictorially in *CGBVS-DNN: Prediction of Compound-protein Interactions Based on Deep Learning*<sup>9</sup>:

---

<sup>9</sup> "CGBVS-DNN: Prediction of Compound-protein Interactions ... - NCBI." Accessed August 14, 2018. <https://www.ncbi.nlm.nih.gov/pubmed/27515489>.



Positive data  
(125 thousands)

```
C00000001 SHT1A_HUMAN
C00000001 ADA2A_HUMAN
C00000001 DRD2_HUMAN
C00000001 ADA2B_HUMAN
C00000001 ADA2C_HUMAN
C00000001 DRD1_HUMAN
C00000001 DRD4_HUMAN
C00000001 DRD5_HUMAN
C00000001 ADA1D_HUMAN
C00000001 SHT1D_HUMAN
C00000001 SHT1B_HUMAN
C00000001 SHT2A_HUMAN
```

(a) Random shuffle

```
C00481854 SHT2A_HUMAN
C01312316 CCR10_HUMAN
C00101743 GRM5_HUMAN
C01414489 SHT2A_HUMAN
C00219713 MTR1A_HUMAN
C00105324 BKRB1_HUMAN
C00570335 MCHR2_HUMAN
C00660536 SHT1F_HUMAN
C00069806 CCR10_HUMAN
C00189409 HRH3_HUMAN
C01370427 DRD4_HUMAN
C00295805 BKRB1_HUMAN
```

(b) Pick out 2 interactions

```
C00481854 SHT2A_HUMAN
C01312316 CCR10_HUMAN
```

(c) Change the combination

```
C00481854 CCR10_HUMAN
C01312316 SHT2A_HUMAN
```

(e) Otherwise

Negative data  
(125 thousands)

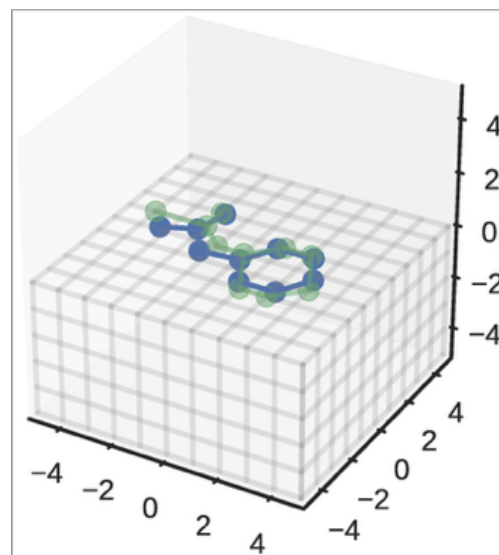
```
C00481854 CCR10_HUMAN
C01312316 SHT2A_HUMAN
C00101743 SHT2A_HUMAN
C01414489 GRM5_HUMAN
C00219713 BKRB1_HUMAN
C00105324 MTR1A_HUMAN
C00570335 SHT1F_HUMAN
C00660536 MCHR2_HUMAN
C00069806 HRH3_HUMAN
C00189409 CCR10_HUMAN
C01370427 BKRB1_HUMAN
C00295805 DRD4_HUMAN
```

(d) No overlap

Figure 1. Generation of negative data.

### 3. Development and evaluation of a deep learning model for protein-ligand binding affinity prediction<sup>10</sup> (Pafnucy Model, 2017)

- a. The **ligand-receptor complex is represented as a 3D grid** (image on right) where the training data is from PDBbind database. The **model treats the atoms of both proteins and ligands in the same manner**. The model tries to predict the **exact binding affinity value** based on this (as opposed to carrying out classification - which causes the loss of information about the actual strength of interaction). This is a **regression** task.



- b. To properly handle orientation of molecular complexes, the **dataset was augmented with systematic rotations of the input data**. This allowed for the **representation of the input as a 4D tensor in which each point is defined by Cartesian coordinates (the first 3 dimensions of the tensor) and a vector of features (the last dimension)**.
- c. **19 features** were used to describe an atom:
- 9 bits (one-hot or all null) encoding atom types: *B, C, N, O, P, S, Se, halogen, and metal*
  - 1 integer (1, 2, or 3) with atom hybridization: *hyb*
  - 1 integer counting the numbers of bonds with other heavyatoms: *heavy\_valence*
  - 1 integer counting the numbers of bonds with other heteroatoms: *hetero\_valence*
  - 5 bits (1 if present) encoding properties defined with SMARTS<sup>11</sup> patterns (SMARTS substitutes a molecule's functional groups with pharmacophoric points ("an ensemble of steric and electronic features necessary for

<sup>10</sup> "Development and evaluation of a deep learning model for protein ...." Accessed August 14, 2018. <https://www.ncbi.nlm.nih.gov/pubmed/29757353>.

<sup>11</sup> "Daylight Theory: SMARTS - A Language for Describing Molecular ...." Accessed August 14, 2018. <http://www.daylight.com/dayhtml/doc/theory/theory.smarts.html>.

molecular recognition of ligand”). Anything you can define with a SMARTS pattern can be a feature, and there are 5 elementary pharmacophoric points already implemented): hydrophobic, aromatic, acceptor, donor, and ring

- vi. 1 float with partial charge: *partialcharge*
  - vii. 1 integer (1 for ligand, -1 for protein) to distinguish between the two molecules: *motype*
- d. In case of collisions (multiple atoms in a single grid point), which rarely occur for a 1-Å grid, features from all colliding atoms were added.
- e. This input tensor hence incorporates the chemical as well as the structural properties of the protein-ligand complex in its entirety.
- f. Interesting results obtained from Pafnucy Model:
- i. The **feature with the widest range is the motype – the feature that distinguishes between the protein from the ligand.**
  - ii. This result implies that **Pafnucy learned that binding affinity depends on the relationship between the two molecules** and that recognising them is crucial.
  - iii. Pafnucy recognises different features of the data in different layers of the neural network. However, the **closer we get to the output layer, the more similar the activations become.**

The figure below represents the activation patterns for each layer and this increase in similarity. It is visible that the **model learned to extract the same information from differently presented data** i.e. Pafnucy managed to give almost identical predictions for two different orientations of the complex (the second rotated about the X-axis by 180° ).

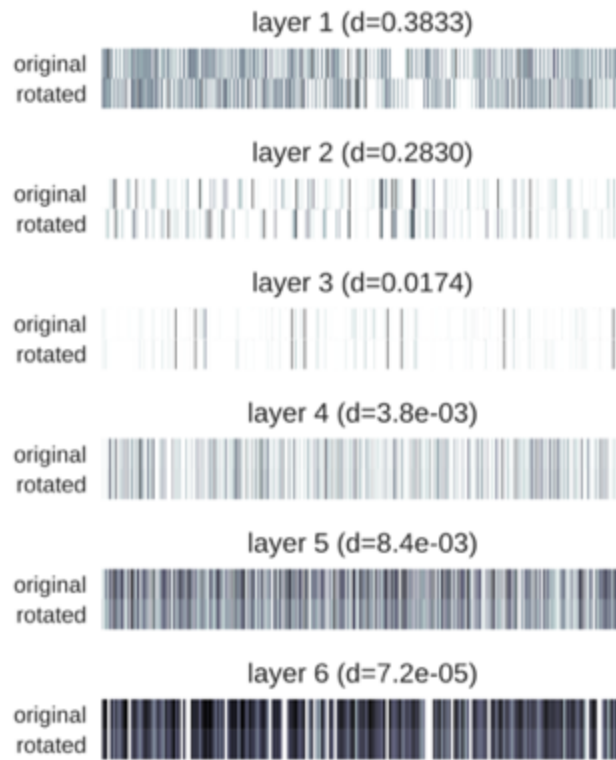


Fig. 5: Activations on the hidden layers for two orientations of the PDE10A complex (PDB ID: 3WS8). Darker colors indicate higher values. Cosine distances ( $d$ ) between the activation patterns for each layer are provided.

## Further Work on Pafnucy Model

The primary model I worked upon for the duration of the project was the Pafnucy Model<sup>12</sup>. The architecture of the model can be represented as follows (image from paper):

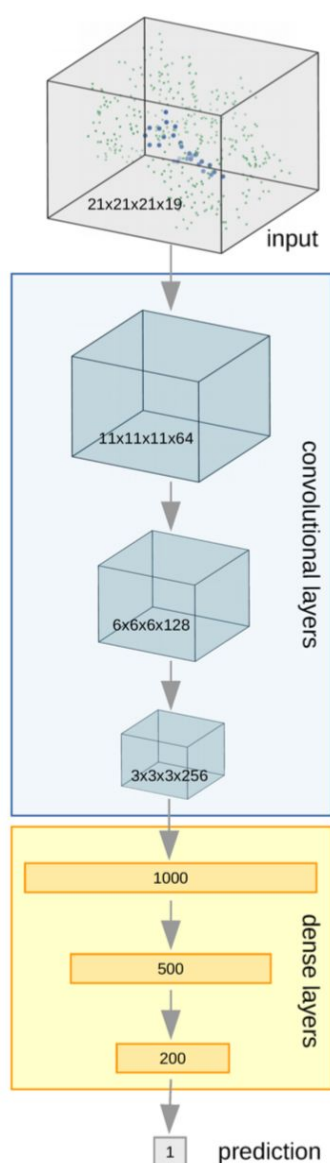


Fig. 1: Pafnucy's architecture. The molecular complex is represented with a 4D tensor, processed by 3 convolutional layers and 3 dense (fully-connected) layers to predict the binding affinity.

<sup>12</sup> "Development and evaluation of a deep learning model for protein ...." Accessed August 14, 2018. <https://www.ncbi.nlm.nih.gov/pubmed/29757353>.

The first step was the preparation of the **PDBbind v2016 Dataset**<sup>13</sup>. This entailed the following:

- The dataset was split into **3 disjoint sets**: the general set (set of all available data), the refined set (set of complexes of higher quality) and the core set (complexes from the refined set were clustered by protein similarity and 5 representative complexes were selected from each cluster)
- The **general and refined sets were used as training sets**, whereas the **core set was used as an external test set**.
- The partitioning of the dataset was carried out using a Jupyter notebook found on the github repository<sup>14</sup> of the authors. This repository also contains code to train the dataset and test it.

The specifications of the model trained are as follows:

- **Three convolutional layers** were used, each having a corresponding max pooling layer. **Max pooling** here served to detect features over broader regions in the 4D input tensor.
- The result of the last convolutional layer was flattened and given as input to **3 dense layers**.
- All convolutional and dense layers used a **ReLU activation function**. ReLU is was chosen by the authors because it sped up the learning process compared to other activation functions they tested.
- An **Adam optimizer** was used for training, with:
  - learning rate =  $10^{-5}$
  - mini-batch size = 5
  - number of epochs = 20
  - **L2 weight decay** used to prevent overfitting, with  $\lambda = 0.001$
- As mentioned earlier, the **dataset was augmented with structural rotations**. This was practically carried out by inputting every structure to the network with 24 different orientations. Quoting from the paper directly:

“By using systematic rotations of complexes during training, we anticipated that the network would learn more general rules about protein- ligand interactions and lead to better performance on new data. Indeed, in our experiments, we observed a much worse

---

<sup>13</sup> "PDBbind." Accessed August 14, 2018. <http://www.pdbbind.org.cn/>.

<sup>14</sup> "cheminflBB / pafnucy · GitLab." Accessed August 14, 2018. <https://gitlab.com/cheminflBB/pafnucy>.

performance of models trained on single orientations regardless of the hyperparameters used to define a particular network.”

The model was run on a CPU (instead of traditional GPUs), which contributed to the increased running time. The Python **conda interface was used to setup the virtual environment required**. While executing, it was observed that after a certain amount of epochs, the backpropagation on the **weights of the neural network would always result in NaN values**.

It was understood that this was a **problem related to exploding gradients**. Exploding gradients can be defined as follows<sup>15</sup>:

“An error gradient is the direction and magnitude calculated during the training of a neural network that is used to update the network weights in the right direction and by the right amount.

In deep networks or recurrent neural networks, error gradients can accumulate during an update and result in very large gradients. These in turn result in large updates to the network weights, and in turn, an unstable network. At an extreme, the values of weights can become so large as to overflow and result in NaN values.”

Issues of exploding gradients can be solved by using **smaller batch sizes in training, using ReLU activations, and using L1 or L2 weight regularization**. Since all these mechanisms were already in practice, two additional approaches were experimented with. Changes were made to source code in the library tfbio<sup>16</sup>, which was a python library written by the authors, that contained important functions and the neural network architecture.

- **He Initialization**

- The paper initially used `tf.truncated_normal_initializer` to initialize the weights of all the layers, with a standard deviation of 0.001.
- This initializer generates a truncated normal distribution with values similar to values from a `random_normal_initializer`. However, any values more than two standard deviations from the mean are discarded and re-drawn<sup>17</sup>.
- However, I used He initialization<sup>18</sup> instead. Here, the **weights are initialized according to the size of the previous layer**. They are still random, but they **differ in range depending on the size of the previous layer of neurons**. This provides a

---

<sup>15</sup> "A Gentle Introduction to Exploding Gradients in Neural Networks." Accessed August 14, 2018.

<https://machinelearningmastery.com/exploding-gradients-in-neural-networks/>.

<sup>16</sup> "cheminflBB / tfbio · GitLab." Accessed August 14, 2018. <https://gitlab.com/cheminflBB/tfbio>.

<sup>17</sup> "tf.truncated\_normal\_initializer | TensorFlow." Accessed August 14, 2018.

[https://www.tensorflow.org/api\\_docs/python/tf/truncated\\_normal\\_initializer](https://www.tensorflow.org/api_docs/python/tf/truncated_normal_initializer).

<sup>18</sup> "Delving Deep into Rectifiers: Surpassing Human-Level Performance ...." Accessed August 14, 2018. <https://arxiv.org/abs/1502.01852>.

faster and more efficient gradient descent, while also reducing possibilities of exploding gradients (since it is a more **controlled initialisation**)<sup>19</sup>.

- **Gradient Clipping**

- Gradient Clipping is a technique to prevent exploding gradients in very deep networks.
- There exist various ways to perform gradient clipping, but the a common one is to normalize the gradients of a parameter vector when its L2 norm exceeds a certain threshold according to:

$$\text{new\_gradients} = \text{gradients} * \text{threshold} / \text{L2\_norm}(\text{gradients})^{20}$$

- What this process achieves is that it **prevents the weights from growing too fast in a single step** and thereby going to NaN state.
- Initially, the part of the code that dealt with the training was as follows:

```
cost = tf.add(mse, l2, name='cost')
optimizer = tf.train.AdamOptimizer(learning_rate, name='optimizer')
train = optimizer.minimize(cost, global_step=global_step,
                           name='train')
```


- To incorporate gradient clipping, it was changed as follows:

```
cost = tf.add(mse, l2, name='cost')
trainable_vars=tf.trainable_variables()
grads, _ = tf.clip_by_global_norm(tf.gradients(cost, trainable_vars), 5)
optimizer = tf.train.AdamOptimizer(learning_rate)
train = optimizer.apply_gradients(zip(grads,
trainable_vars),global_step=global_step, name='train')
```

<sup>19</sup> "Initialization Of Deep Networks Case of Rectifiers | DeepGrid." Accessed August 14, 2018. <http://www.jefkine.com/deep/2016/08/08/initialization-of-deep-networks-case-of-rectifiers/>.

<sup>20</sup> "Deep Learning Glossary – WildML." Accessed August 14, 2018. <http://www.wildml.com/deep-learning-glossary/>.





Working with these two methods, and additional adjustment of batch sizes, however, did not resolve the problem of exploding gradients in the code. This issue would need to be fixed to practically understand the effectiveness of the Pafnucy model and experiment with it.

## Scope in The Future

Some parts of the Pafnucy model that can be explored and extended further in the future:



- **Fixing the problem of exploding gradients:** The authors do not report this issue on their github repository. However, as this issue was repeatedly encountered in the duration of the project, to advance any further, it needs to be fixed.
- This model **does not include any topological descriptors**, as seen in previously reviewed papers like CGBVS-DNN. It was also observed in **TopologyNet**<sup>21</sup>, which was noted as an interesting architecture to review in the future. The advantages of topological descriptors in detecting structural and chemical features have been mentioned in the literature review for CGBVS-DNN, in this report.
- Because Pafnucy is a neural network, it is also **possible to calculate and analyze its gradients**. During training, gradients are used to optimize model parameters. However, they can also be **calculated for the input and point to beneficial changes in a molecule's conformation (finding optimal pose during docking) or composition (lead optimization)**.
- Towards the end of this project, a paper titled **Visualizing convolutional neural network protein-ligand scoring**<sup>22</sup> was published by Hochuli et al. It detailed several methods by which the black-box mechanism of neural networks could be 'unboxed' in some way.
  - The authors attempted to **visualise what patterns each layer was learning** in a CNN architecture they had devised earlier<sup>23</sup>.
  - They carried out the following:
    - measured the **sensitivity of the model to changes in the input**,
    - attempted to **visualise the gradients of backpropagation**,

---

<sup>21</sup> "TopologyNet: Topology based deep convolutional and multi ... - PLOS." Accessed August 14, 2018. <http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1005690>.

<sup>22</sup> "Visualizing Convolutional Neural Network Protein-Ligand Scoring." Accessed August 14, 2018. <https://arxiv.org/abs/1803.02398>.

<sup>23</sup> "Protein–Ligand Scoring with Convolutional Neural Networks - Journal ...." Accessed August 14, 2018. <http://pubs.acs.org/doi/abs/10.1021/acs.jcim.6b00740>.

- 
- used **Layer-wise Relevance Propagation** (LRP)<sup>24</sup> to map classification probabilities back to the original input
  - These methods can **hypothetically be applied to Pafnucy as well**, to understand how the 3D complexes are interpreted by the model. These visualisations can **influence future changes to network design**, and the **design of a new newtork** as well.
- 

---

<sup>24</sup> "Layer-wise Relevance Propagation for Neural Networks with Local ...." Accessed August 14, 2018. <https://arxiv.org/abs/1604.00825>.