CS221 Project Progress Report

Our project task is to predict binding affinities of modified bis-guanidinium neurotoxins **(the ligand)** to different isoforms of voltage-gated sodium ion channels **(the protein)**. To review, our training data is a set of 300 experimentally collected binding affinities of ligands to proteins. An example datapoint is as follows:

Toxin Subclass	Compound (Common Name)	Compound (SMILES)	Protein	Protein DEKA Sequence	Binding Affinity (IC50)	Unit	pKi	Ki	Potent	Source
STX	STX.N21.1	[NH2+]=C1N[C@@H](OC(NC)=O)[C@H](NC2=[NH2+])[C@]3(N2)N1CCC3(O)O	rNaV1.4	TQDYWENCGEWIETTFKGWMDTSAGWDG	1.9	nM	0.02228	0.95	1	RTT

“Toxin subclass” refers to the scaffold on which the ligand was created. There are four scaffolds in our dataset (STX, ZTX, GTX, TTX); each is a different bis-guanidinium natural product. The “Compound (Common Name)” column stores the ID of the ligand. It maps to a seperate file with 2-D line drawings of the ligand for easy visualization. There are about 40 unique ligands in our dataset. The “Compound (SMILES)” column stores a string representation of the ligand. The SMILES string encodes all the information necessary to visualize the ligand in three dimensions (such as the atomic composition, bonding and structure). The “Protein” column stores the common name of the protein. All proteins in this dataset are isoforms or mutated isoforms of rat or human sodium channels; there are about 70 unique proteins in our dataset. Because all our ligands are based on bis-guanidinium neurotoxins that are hypothesized to restrict the access of sodium ions to the protein by plugging up the protein above the sodium selectivity filter like a cork, modifications made to the protein sequence are concentrated in the DEKA region, a loop of amino acids bordering the point in the protein channel where toxins are hypothesized to dock. “Protein DEKA Sequence” stores the amino acids present in this region; the composition of amino acids in this region uniquely identifies each of the 70 proteins in our dataset. “Binding affinity” stores the experimentally measured IC50 in nanomolars of the specific ligand/protein pair identified by the row. “pKi” and “Ki” store the inhibition constant, another measure of binding affinity. Although this value can be experimentally measured, it was not in our case and was rather calculated by making the rough approximation that Ki = ½*(IC50) for the case of competitive protein-ligand interactions. Since all ligands in our dataset competitively restrict the access of sodium ions into the protein, this approximation is a fair assumption to make. “Potent” is a binary 0/1 label; ligands are deemed potent if their IC50 is less than or equal to 500 nM. Finally, “Source” maps to the researcher that collected the experimental datapoint; all datapoints were gathered by hand by this project group from theses written by members of the Du Bois group at Stanford.

To summarize, our problem input is a protein/ligand pair, and our desired output is a measure of binding affinity. The algorithm/model we would like to use to solve our problem is linear regression passed along layers of a pre-trained convolutional neural network called Pafnucy. Pafnucy [paphnusy] is a 3D convolutional neural network developed by Marta Stepniewska-Dziubinska that predicts binding affinity for protein-ligand complexes. It was trained on the PDBbind database and tested on the CASF "scoring power" benchmark.

Protein-ligand complexes must be transformed and encoded into tensors in order to be utilized by a neural network. Pafnucy accomplishes this by cropping three-dimensional coordinate files of a protein-ligand complex to a defined 20-Å cubic box centered at the geometric center of the ligand. Positions of heavy atoms are discretized using a 3D grid with 1-Å resolution, allowing for each input to be represented as a 4D tensor in which each point (atom) is defined by its Cartesian coordinates (the first three dimensions of the tensor) and a vector of features (the last dimension). Inputs to Pafnucy use the following 19 features in its last dimension to describe an atom:

- 9 bits (one-hot or all null) encoding atom type: B, C, N, O, P, S, Se, halogen and metal
- 1 integer (1, 2, or 3) storing atom hybridization.
- 1 integer storing the number of bonds with heavy atoms.
- 1 integer storing the number of bonds with heteroatoms.
- 5 bits encoding properties like hydrophobicity, aromaticity, H-bond acceptor/donor status, and ring presence.
- 1 float storing partial charge.
- 1 integer (1 for ligand, -1 for protein) to distinguish between the two molecules.

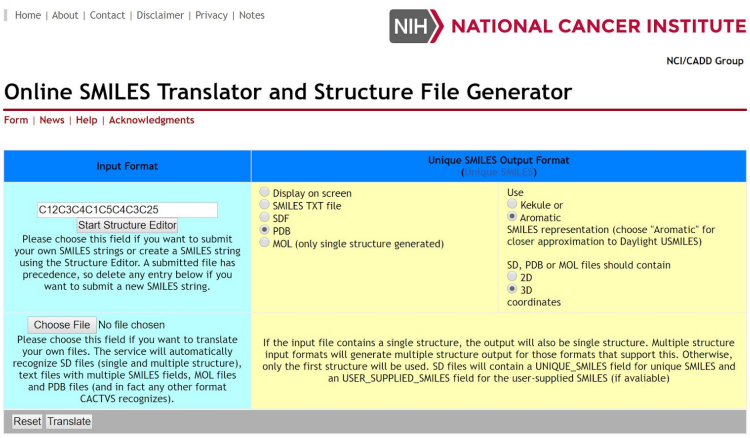
Pafnucy was trained and tested on protein-ligand complexes from the PDBbind database v. 2016. This database contained 3D structures of molecular complexes and their corresponding binding affinities expressed with pKa (-logKi) values. All complexes were protonated and charged using UCSF Chimera with Amber ff14SB for standard residues and AM1-BCC for non-standard residues and ligands. About 11,906 complexes total were used in Pafnucy's training set; 1000 complexes were used in validation, and 290 complexes were used as an external test set. The training, test, and validation sets were all disjoint.

Pafnucy's architecture is a deep convolutional neural network with a single output neuron for predicting binding affinity. The input is first processed by a block of 3D convolutional layers. Pafnucy uses 3 convolutional layers with 64, 128, and 256 filters. Each layer's filters have 5-Å cubic resolution and are followed by a max pooling layer with a 2-Å cubic patch. The result of the last convolutional layer is flattened and used as input for a block of dense layers. Pafnucy uses 3 dense (fully connected) layers with 1000, 500, and 200 neurons. Dropout with drop probability of 0.5 was used for all dense layers. Both convolutional and dense layers are composed of ReLU units.

In the training stage, Pafnucy was initialized with values for the convolutional filter weights drawn from a truncated normal distribution with mean 0 and standard deviation 0.001. Biases were set to 0.1. The weights in the dense were initialized with a truncated normal distribution with mean 0 and standard deviation $1/\sqrt{n}$, where n was the number of incoming neurons for a given layer. The biases for the dense layers were set to 1.0. The network was trained using the Adam optimizer with a learning rate of 10^{-5} and 5 examples per minibatch. Training was carried out for 20 epochs, but the model performance was optimized after 14 epochs of training. Dropout and L2 weight decay ($\lambda = 0.001$) was implemented to reduce overfitting. Every input structure was presented to the network in 24 different orientations to develop a model that is not sensitive to ligand-receptor complex orientation. Thus, 24 training examples are generated for each protein-ligand complex.

We want to perform transfer learning on Pafnucy using the dataset of 300 neurotoxin-sodium channel binding pairs. Our process of featurizing the dataset we had gathered to create input for Pafnucy is as follows:

First, we wrote a script to convert the ligands SMILES strings to 3D structure files with the .pdb (Protein Data Bank) file extension using the online SMILES translator developed by the NHI (https://cactus.nci.nih.gov/translate/):



Next, we generated .pdb files for each one of our 70 unique sodium channels. Starting from a cryo-EM image of human sodium channel 1.7 (hNaV1.7) bound to saxitoxin with 3.2-Å resolution, we extracted the coordinates of the protein and

identified the amino acids forming the DEKA loop. Next, we mapped hNaV1.7's DEKA loop onto rat sodium channel 1.4's DEKA loop by feeding their respective amino acid sequences to BLAST (the Basic Local Alignment Search Tool). We also mapped each of the rat sodium channels 1.1-1.9 to the corresponding human isoform.

Having acquired the sequence correspondence code, we wrote a script to edit the .pdb file of hNaV1.7. Our script modifies amino acids in the DEKA loop by editing the coordinates and atomic identities of the relevant sidechains. The figure below shows PyMol's visualization of the structure file of hNaV1.7 (our template). The DEKA region is in the center of the protein. To the right is a text representation of the same structure file with the amino acid at position 1410 highlighted. The regions in green correspond to the backbone atoms, which remain the same for each amino acid. The regions in red represent the side chain atoms, which change depending on amino acid identity. Side chains are attached at the atom labeled 'CA' in the third column of the green region; thus, by treating the CA atom as the zero-coordinate and adding/removing atoms from the red region as appropriate, we can edit the PDB file to generate three-dimensional structure files of the proteins for which we don't have cryo-EM images. We generated theoretical three-dimensional coordinates for new side chains by referencing the geometry of amino acids in vacuo.

```
ATOM   6308  CH2 TRP A1408     142.149 137.344 137.799  1.00 23.41           C
ATOM   6309  N   THR A1409     143.099 133.510 142.648  1.00 25.11           N
ATOM   6310  CA  THR A1409     143.746 134.492 143.509  1.00 25.11           C
ATOM   6311  C   THR A1409     144.597 133.916 144.616  1.00 25.11           C
ATOM   6312  O   THR A1409     145.611 134.489 144.949  1.00 25.11           O
ATOM   6313  CB  THR A1409     142.792 135.567 144.054  1.00 25.11           C
ATOM   6314  CG2 THR A1409     141.668 135.824 143.087  1.00 25.11           C
ATOM   6315  OG1 THR A1409     142.261 135.149 145.310  1.00 25.11           O
ATOM   6316  N   ILE A1410     144.228 132.770 145.231  1.00 23.64           N
ATOM   6317  CA  ILE A1410     145.044 132.197 146.298  1.00 23.64           C
ATOM   6318  C   ILE A1410     146.408 131.844 145.738  1.00 23.64           C
ATOM   6319  O   ILE A1410     147.412 132.039 146.396  1.00 23.64           O
ATOM   6320  CB  ILE A1410     144.434 130.932 146.914  1.00 23.64           C
ATOM   6321  CG1 ILE A1410     142.951 131.125 147.188  1.00 23.64           C
ATOM   6322  CG2 ILE A1410     145.154 130.580 148.202  1.00 23.64           C
ATOM   6323  CD1 ILE A1410     142.208 129.819 147.316  1.00 23.64           C
ATOM   6324  N   ILE A1411     146.405 131.289 144.526  1.00 23.81           N
ATOM   6325  CA  ILE A1411     147.650 130.925 143.861  1.00 23.81           C
ATOM   6326  C   ILE A1411     148.541 132.125 143.545  1.00 23.81           C
ATOM   6327  O   ILE A1411     149.731 132.078 143.771  1.00 23.81           O
```

Having created three-dimensional structure files for ligands and proteins, we proceeded to dock the ligands against the proteins to create ligand-receptor complex inputs for Pafnucy. To do so, we used Vina, the current industry standard for protein-ligand docking. Vina uses molecular mechanics simulations combined with multithreading to predict the geometry and binding affinity of small molecule ligands to protein pockets. To use Vina, we first wrote scripts to convert the PDB files of our ligands and proteins to PDBQT files using Vina's companion software, AutoDock. The PDB->PDBQT conversion process adds polar hydrogen atoms and partial charges to the protein and ligand structure files. Having generated PDBQT files for all our protein and ligand pairs, we wrote a script to push the data through Vina. Vina takes a configuration file that specifies the search space for the docking program, which we defined as a 20-Å cubic box oriented at the geometric center of the DEKA loop. Vina's output is a pdb file of ligand docking poses along with a predicted binding affinity (in kcal/mol). We selected the highest affinity result as our hypothesized docking pose and stored the binding affinity that Vina output as our oracle.

Our final step was to overlay the protein structure files over our ligand docking file, to combine the two objects, and to extract a 20-Å cubic box from the protein, centered at the geometric center of the ligand, to offer to Pafnucy as input. We accomplished this by writing a Python script in PyMol. We ran the pocket structure file and ligand structure file through Pafnucy's featurizer to create the 4D tensors described previously. Then, we fed the input to Pafnucy to output numbers for predicted binding affinity.

Our project baseline was a unigram naive bayes classifier using SMILES strings and protein amino acid sequences from the DEKA loop as features, which achieved a classification accuracy of 47% on our training dataset (using an 80:20 train/test split). Note that we trained a classification, not a regression algorithm for our baseline, so the baseline accuracy cannot be directly compared against the oracle accuracy as is. Our project oracle is Vina's binding affinity output, which we converted to Ki using the following equation, where ΔG is Vina's output, T=298K, and R=1.986 cal/mol K.

$$Ki = exp(\Delta G/RT)$$

We have yet to fully automate this process for our entire dataset, but we were able to run our baseline algorithm for a small subset of the data.

The oracle (Vina) achieved a percent error (defined as 100%*[observed binding affinity - expected binding affinity]/expected binding affinity) of -302%.

In comparison, our initial algorithm (a single run through Pafnucy) achieved a percent error of -327%.

The output generated by Pafnucy compares favorably to the output generated by the oracle (they match almost exactly). This is good because it suggests that our initial algorithm is already very well-adapted towards solving this problem, and it is a good template from which to optimize further. However, both Pafnucy and the oracle's generated binding affinity predictions are far off from the actual experimental Ki's. This may actually be a problem with our data translation process. Recall that we were able to glean only IC50's from our initial dataset, which we translated to Ki's via a rough approximation that Ki = IC50/2. The actual conversion process involves a more rigorous consideration of Michaelis-Menten kinetics, a calculation that depends on the concentration of the protein and ligands in solution. We will work on gathering the data to perform this calculation as we continue to develop this project; hopefully, this will correct our values moving forward.