The original paper “*BionoiNet: Ligand-Binding Site Classification with Off-the-Shelf Deep Neural Networks”* takes the three dimensional structure information from different small proteins and peptides and reduces the dimensionality of these structures to a 2D representation.

They used a pretrained ResNet18 model (a CNN model) (do more research about this baseline model) and 2D Voronoi diagrams to classify ligand binding sites. They do feature extraction of the ResNet meaning they don’t train the model from scratch, but rather they freeze the layers from the ResNet and retrained on the last layer to predict their classes.

Their logic states that there are already multiple models that predict the binding pockets of molecules but these are this dimensionality reduction reduces the complexity of the model and eases the computational power needed to generate good predictions: here is more information about it: Nonetheless, BionoiNet has four major advantages over 3D methods. First, 2D images are more efficient in terms of computing time, memory and storage compared to 3D voxels. For example, calculating a 3D voxel requires 10–30 min on a single core and produces a binary file whose size is 3.7 MB. In contrast, Voronoi images are only about 14 kB in size and can be generated in a fraction of second, which is highly advantageous when working with large datasets

Their project focuses on predicting the ligand binding site of nucleotides and heme-binding pockets. This is how they made sure that the 3D information was not lost in the 2d plane: Next, the Miller cylindrical projection, originally devised to portray

the Earth in two dimensions (Miller, 1942), is computed to

generate the 2D coordinates of binding site atoms. Given that binding

sites typically have irregular structures, this particular projection

was found to maintain relative distances between atoms so that the

spatial arrangement of atoms is preserved after projection.

Furthermore, the Miller projection avoids eclipsing atoms, i.e. projecting

atoms from the opposite sides of a pocket next to one another

on the 2D plane. The projected atoms are used as seeds for a

Voronoi tessellation

Their model does binary prediction, it determines if a binding site is Nucleotide binding site or Heme binding site.

The other paper that I used for my project was: Bacterial Immunogenicity Prediction by Machine Learning Methods. From this paper they tested multiple models like k neasest Neighbor, SVM. Random Forest and XgBoost. Their dataset was extracted from the VaxiJen Server to predict only Bacterial Immunogenicity using something called “E-Descriptors” which are a set of numerical values that describe its physicochemical properties for example: A (Alanine): [0.62, 0.50, 0.36, 0.23, 0.18]

**Hydrophobicity, Size**, **α-helix propensity**, **Number of codons per amino acid** (reflecting genetic code redundancy), **β-strand frequency**

Their best performing model was the RSM-1NN with an accuracy of 0.79 on the training set and 0.82 on the test set.

So far my project aims to use from the same database they used, the proteins present in cancer to predict tumor immunogenic reactions and non-tumor inmmunogenic reactions, but using the Voronoi tessellation approach as I believe the structural information can provide additional relevant features to the antigenic prediction, then exand this from binary classification of the protein (tumor proteins as immunogenic vs non-immunogenic proteins) to the multiclass classification (tumor immunogenic, bacterial immunogenic, tumor non-immunogenic and bacterial non-immunogenic.)

First thing is to:

1. Preprocess the datasets to convert from Fasta files to Voronoi tessellation.
2. Preprocess the dataset to convert the FASTA files to Graph Neural Networks.

The folders are split in the following way:

1. Tumor Antigens immunogenic (521) is the same as Tumor Immunogenic
2. Tumor Antigen non-immunogenic (520) is the same as Tumor Non-Immunogenic
3. Bacterial antigen (216)
4. Bacterial non\_antigens (314)

Models that need to be developed are:

1. For tumor dataset:
   1. Benchmark Model of IBionet (ResNet 18 with feature extraction) binary classification
   2. Benchmark Model of IBionet (ResNet 18) but with retraining binary classification
2. Graph Neural Network for binary classification tumor proteins
3. Increased depth in Graph Neural Networks (two more hidden layers) tumor proteins
4. Third Graph Neural Network (even deeper) gnn2 which lowers the learning rate to (0.003) and decreases the number of epochs to 30

The models at this point were struggling to learn with the Graph Neural Networks.

Because the GNN did not perform well:

Epoch 30/30 - Loss: 12.8415, Train Acc: 0.7445, Val Acc: 0.7647, Val Loss: 0.5048 in 114 minutes. And the model plateaud basically somewhere around epoch 9. Although we do see that the GNN does not overfit as the training accuracy is the same as the validation accuracy.

I then tried a CNN to see if it learns features better or if the only reason the ResNet has performed better than the GNN was because it is a very deep architecture, and doesn’t take long to learn because it is transferred learning.

1. Custom-made binary classification CNN for tumor proteins.

Based on the CNN architecture results (Epoch 10/10 - Loss: 0.7754, Train Acc: 0.9987, Val Acc: 0.7647, Val Loss: 0.6337) we can see that we don't need a very complex architecture as Resnet to get very similar results (Resnet Results were: Epoch 30/30 - Loss: 0.3731, Train Acc: 0.9949, Val Acc: 0.8235, Val Loss: 1.0171). The simple CNN is actually overfitting to the training set. Given that the CNN was overfitting I though making it multiclass would increase the dataset (sort of data augmentation) because the would now be more samples that are not tumor specific but that still have protein properties and also immunogenic/non-immunogenic properties. Now we will escalate the previous CNN but for multiclass classification. We wil add bacterial antigens and bacterial non antigens.

Now the dataset size was: Train Samples: 1175, Val Samples: 78, Test Samples: 314.

This was the dataset distribution:

Training samples: 1182

Class 0: 386

Class 1: 409

Class 2: 233

Class 3: 154

Validation samples: 78

Class 0: 29

Class 1: 22

Class 2: 17

Class 3: 10

Test samples: 316

Class 0: 113

Class 1: 89

Class 2: 63

Class 3: 51

So even though it was not perfectly distributed there was no extreme overrepresentation of any class. The model again even overfit to the new dataframe giving us very similar results:

Epoch 10/10 - Loss: 0.9682, Train Acc: 0.9957, Val Acc: 0.7051, Val Loss: 0.8331