**Università degli Studi di Padova – Department of Mathematics**

**Master’s Degree in Data Science**

**Structural Bioinformatics, A.Y. 2022/23**

Tanner Graves 2073559 tanneraaron.graves@studenti.unipd.it

Marco Uderzo 2096998 marco.uderzo@studenti.unipd.it

Nour Alhousseini 2081230 nour.alhousseini@studenti.unipd.it

Hazeezat Adebimpe Adebayo 2090254 hazeezatadebimpe.adebayo@studenti.unipd.it

­­­

Classification of Contacts in Protein Structures

**1.** Training Set and Data Retrieval

The first step of the project was retrieving the data from each .tsv file from the features\_ring folder (1,807 PDBs in total) and storing it into a single DataFrame.

The data contained in each .tsv file consists in a tab-separated file available for each protein, in which each row represents a contact in the protein and each column represents a feature about that contact. The last column is the target label, specifying the interaction type.

|  |  |  |  |
| --- | --- | --- | --- |
| **Column position** | **Column name** | **Column meaning** | **Type of column** |
| 1 | **pdb\_id** |  |  |
| 2 | **s\_ch** | chain | source residue identifier |
| 3 | **s\_resi** | index |
| 4 | **s\_ins** | insertion code |
| 5 | **s\_resn** | name |
| 6 | **s\_ss8** | secondary structure 8 states (DSSP) | source residue features |
| 7 | **s\_rsa** | relative solvent accessibility |
| 8 | **s\_up** | half sphere exposure up |
| 9 | **s\_down** | half sphere exposure down |
| 10 | **s\_phi** | phi angle |
| 11 | **s\_psi** | psi angle |
| 12 | **s\_ss3** | secondary structure 3 states (from angles) |
| 13 | **s\_a1** | Atchley feature 1 |
| 14 | **s\_a2** | Atchley feature 2 |
| 15 | **s\_a3** | Atchley feature 3 |
| 16 | **s\_a4** | Atchley feature 4 |
| 17 | **s\_a5** | Atchley feature 5 |
| 18 | **t\_ch** | chain | target residue identifier |
| 19 | **t\_resi** | index |
| 20 | **t\_ins** | insertion code |
| 21 | **t\_resn** | name |
| 22 | **t\_ss8** | secondary structure 8 states (DSSP) | target residue features |
| 23 | **t\_rsa** | relative solvent accessibility |
| 24 | **t\_up** | half sphere exposure up |
| 25 | **t\_down** | half sphere exposure down |
| 26 | **t\_phi** | phi angle |
| 27 | **t\_psi** | psi angle |
| 28 | **t\_ss3** | secondary structure 3 states (from angles) |
| 29 | **t\_a1** | Atchley feature 1 |
| 30 | **t\_a2** | Atchley feature 2 |
| 31 | **t\_a3** | Atchley feature 3 |
| 32 | **t\_a4** | Atchley feature 4 |
| 33 | **t\_a5** | Atchley feature 5 |
| 34 | **Interaction** | interaction type |  |

Table 1.1: Training Set Description

|  |  |
| --- | --- |
| **Interaction Type** | **Count** |
| HBOND | 333,346 |
| VDW | 155,789 |
| PIPISTACK | 10,403 |
| IONIC | 9,068 |
| SSBOND | 866 |
| PICATION | 626 |
| *Unclassified* | 225,412 |

Table 1.2: Number of Examples by Bond Type.

2. Data Preprocessing

The preprocessing pipeline starts with removing all samples where the label is unavailable (to explain how we are reinputting them).  Then, the missing values of each feature are replaced using the mean of the feature itself. Only numerical features are used.

It was considered to perform *Best Subset Selection* using Logistic Regression to determine which features were most meaningful and the ones that were not influencing the decision much. However, by doing that, the overall performance dropped, so it was decided to discard this approach and keep all features for training.

Scaling is then performed to standardize all the features to values between [0,1] to be then fed to the model.

The biggest criticality in the dataset is the heavy imbalance that is evident by looking at the number of contacts by interaction type. Notably, *Hydrogen Bonds* (HBOND) and *Van der Waals Contacts* (VDW) are the most numerous, which makes them overrepresented in the training set. The remaining contact types, instead, are underrepresented. Training a model with such unbalanced datasets is sure to yield poor performance, especially when evaluating the model on new unseen data.

To mitigate this issue, oversampling the underrepresented classes is applied to encourage the model to better learn the minority interactions. It is to be noted, however, that altering the dataset, especially by oversampling, can yield overly optimistic results in the training performance, which don’t necessarily transfer to as good performances at inference time on new data. Therefore, the balancing process of the training set is performed carefully and conservatively.

The use of undersampling lead to greatly diminished performance distinguishing the majority classes (Hydrogen Bonds and VDW Interactions). For oversampling, *SMOTE* (Synthetic Minority Oversampling TEchnique) is used, which uses interpolation between samples to create new artificial data points. The sampling strategy is {IONIC: 20’000, PIPISTACK: 10’000, PICATION: 20’000, SSBOND: 10’000}

3. Model

3.1. Deep Neural Network

The model created to classify residue-residue contacts is a *Deep Neural Network* for multiclass classification. The Deep Learning library of choice is *Keras*, a very commonly used open-source library that acts as an interface for *TensorFlow*.

The first step to be carried out is the encoding of the label of each sample into an identity vector, using a common practice called “*One-Hot Encoding*”. This ensures that each class is uniquely identified and independent of the others. It helps the neural network to better understand the categorical nature of the data and prevents any ordinal relationship assumptions between the classes.

*One-Hot Encoding* is also performed to then set the number of neurons of the output layer to be equal to the number of classes in the dataset, in order to train each output neuron to determine the probability P(*Ci*| data), with ∑ P(*Ci* | data) = 1. Therefore, the output layer represents the probability distribution of a contact being of a certain type.

Although the model is a relatively small Deep Neural Network with fully connected layers, overfitting is mitigated using *L2 Regularization* *(Weight Decay)* on each Dense layer, and *Early Stopping* monitoring the loss progression over the epochs. These two techniques provide better generalization performance. The use of random dropout of neurons after each hidden layer was also tested, without any meaningful improvement on final test performance.

3.2. Model Details and Hyperparameters

|  |  |
| --- | --- |
| **Hyperparameters** | |
| Batch Size | 16’000 |
| Weight Initialization | Xavier (GlorotNormal) |
| Loss Function | Categorical Cross-Entropy |
| Optimizer | Adam |
| Hidden Layers: Activation Function | ReLU |
| Output Layer: Activation Function | Softmax |
|  |  |

Table 3.2.1: Model’s hyperparameters (Keras)

|  |  |  |
| --- | --- | --- |
| **Layer Type** | **Output Shape** | **# Param** |
| Dense | 128 | 2432 |
| Dense | 128 | 16512 |
| Dense | 128 | 16512 |
| Dense | 128 | 16512 |
| Dense | 128 | 16512 |
| Dense | 6 | 774 |
| # Total Parameters |  | 69,254 |

Table 3.2.2: Model’s architecture (Keras)

4. Results

4.1 Performance

Testing data consisted of 51,608 examples, or 10% of unbalanced non-preprocessed data. Particular attention was put into not having any leak of the oversampled training set data into the test set. well behaved loss (*figure 4.1*) and AUC (*figure 4.2*) histories are evidence of successful model improvement. As demonstrated by the accompanying metrics, the model preforms better that the provided Naive Bayes classifier, achieving *on the test set* a Total Accuracy of 71%, ROC AUC of 0.79 (*figure 4.2*), and a Matthews Correlation Coefficient of 0.38.

A graph of a training and validation loss

Description automatically generated

Figure 4.1: Training and Validation Loss

A graph with red and blue lines

Description automatically generated

Figure 4.2: Training and Validation AUC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| HBOND | 0.77 | 0.85 | 0.81 | 33656 |
| IONIC | 0.46 | 0.26 | 0.33 | 916 |
| PICATION | 0.05 | 0.60 | 0.09 | 63 |
| PIPISTACK | 0.39 | 0.99 | 0.56 | 1050 |
| SSBOND | 0.43 | 1.00 | 0.60 | 88 |
| VDW | 0.61 | 0.41 | 0.49 | 15735 |
| macro avg | 0.45 | 0.68 | 0.48 | 51508 |
| weighted avg | 0.71 | 0.70 | 0.70 |

Table 4.1: Test Performance by Interaction Type

4.2 Issues

In attempt to address poor precision on minority classes, many alternative approaches were tested, including:

- Ensemble Methods: One-vs-One, One-vs-All, and Model Stacking

- Different architectures

- Regularization techniques: Dropout, L2, Batch Normalization, Class Weighting

- Balancing

Nonetheless, it was found that ordinary Deep Neural Networks had the best compromise between minority class and majority class precision. Models that showed improvements in one of these areas consistently suffered in the other.

Implementing dataset balancing greatly improved model recall on minority classes. This would indicate that the model is effectively learning information about the classes. However, these classes still suffer from poor precision with resampling likely attributed to the relatively extremely low unique examples not allowing for a good representation of the class to be learned. The model additionally showed great sensitivity to undersampling, quickly leading to diminished ability to distinguish *Hydrogen Bonds* and *VDW Interactions*, the largest influence on total accuracy. The final model relied primarily on having the minority classes oversampled. Though, oversampling additionally quickly lead to poor generalization on minority classes due to the extremely low number of unique examples making learning a good representation of the classes difficult.

5. Usage

To predict the contacts of a new PDB file using the pretrained model, run the following command in your terminal, using arguments:

> python3 contact\_net.py --predict --pdb your\_pdb\_id

> python3 contact\_net.py --predict --pdb your\_pdb\_id --full

To perform retraining or the model, instead, run:

> python3 contact\_net.py --train

The full documentation of the software is available in the GitHub repository at the following [link](https://github.com/TannerAGraves/StructBioInfo/blob/main/docs/documentation.md)

5.1 Example and Comparison with RING

An illustrative demonstration of usage can be seen with comparing the outputs of the model with the output of RING for *Deoxy Human Hemoglobin* (1A3N):

python3 contact\_net.py --predict --pdb 1a3n --full

↳ generates /data/output/1a3n\_pred.csv

Submitting a corresponding 1a3n.pdb to RING allows us to compare the outputs of the two models (*table 5.1.1*). Note that because the output of calculate\_features.py and RING will provide multiple classifications for the same contact, the mode will be taken for each so accuracy can be calculated. Because the example of *Deoxy Human Hemoglobin* is a well-ordered globular protein, we can see that the model here provides favourable results, delivering an overall accuracy of 80 % on the one protein.

A table of numbers and letters

Description automatically generated

Table 5.1.1: Comparison between ContactNet Predictions and RING.