# Project 2. Single amino acid prediction at protein-protein interaction interfaces

Max Jansen, Karol Roman, Jack Zeng
*Machine Learning Course CS-433, EPFL, Switzerland*

***Abstract*—Proteins consist of building blocks which are called amino acids. Computational protein design can benefit from accurate amino acid type predictions at the interfaces between proteins. In this paper, convolutional neural networks, were applied to predict the an amino-acid type on one side of a protein-protein interaction, using only information from the opposing protein surface. The results of the machine learning scheme are compared to those of RosettaSurf [1]. While inferior, the results provide a promising start. Furthermore, our approach uses a limited amount of data (only from the opposing protein) to make predictions and is thus expected to be more challenging.**

## I. Introduction

Proteins are macro-molecules consisting of a relatively small set of molecular building blocks, called amino acids, arranged in long, highly specific sequences. Once an amino acid becomes part of the chain that forms a protein, it is referred to as a residue. Proteins are versatile in their ability to perform biological functions because of the variety of chemical traits of these amino acids. Some examples of protein functions include cell signaling, immune response, metabolism and structural functions in cells and tissues. The majority of proteins interact with other proteins for larger biological systems to perform these functions. A pair of proteins that interact are called a protein-protein interaction (PPI).

### *A. Protein design*

The ultimate purpose of amino acid recovery methods presented in this paper is to optimise the binding and design of novel proteins. The emphasis would be on the engineering of interfaces with high affinity between these and other proteins. The field of molecular biology has already seen a rapid improvement in the sequencing DNA, oligonucleotide synthesis, protein structure prediction, and *de novo* protein design [2]. These advancements can advance many industries, ranging from healthcare to food production.

Before moving on to engineering, one should first attempt to recover masked residues at the interface between bound proteins. In our case, on naturally occurring pairs of proteins. An algorithm that can do this reliably should also be able to predict which residues would be optimal in novel proteins. Although protein structure prediction is largely solved, the field has had more difficulty computationally designing proteins that bind to their intended target molecules. Much progress in the field was achieved using the Rosetta suite [3]. The Rosetta model uses an energy function that approximates physical optima to predict whether certain configurations are favorable. In spite of the progress, it is still relatively slow and is not yet accurate enough to guarantee that a predicted modification will lead to the intended biological result.

Another category of tools for computational protein design leverage properties of the protein surface using machine learning models [4], [5]. The main idea being that what matters most for interactions of proteins with other molecules are the geometric and physio-chemical parameters of the surface. By abstracting away the interior of the protein, the aforementioned methods can quickly predict multiple aspects of protein interaction.

### *B. Machine learning for single amino acid prediction at PPI interfaces*

The aforementioned tools for computational protein design do not have to be mutually exclusive, however. Most importantly, one can use the opposing surface to inform binder sequence design. Rather than optimizing the entire designed binder sequence or surface to increase affinity, we envision an approach which optimizes a single residue at a time. The reasons we believe our approach of optimizing single amino acids based on opposing surfaces can accelerate protein design are twofold. Firstly, the dMaSIF [5] approach has shown that informative surfaces can be generated in less time than non-machine learning based methods. Secondly, given a binder that is already reasonably compatible with a target, one can accelerate the optimization of its design even further by restricting the computation to a single residue at a time.

One can take two proteins involved in a PPI and compute the (dMaSIF) surface points for one protein while trying to predict residues on the opposing interface. Instead of taking whole surfaces to predict a given residue, one can limit the "field of view" of a residue to a patch of the $N$ nearest surface points.

Amino acids consist of a limited set of atoms common in organic molecules. To describe a protein, their coordinates are commonly stored in the PDB-filetype. Throughout this report we shall refer mostly to the backbone $C_\alpha$ atom and the $C_\beta$. These carbons are present in almost all amino acids types and hence form a useful spatial reference point for selecting patches and calculating distances and angles.

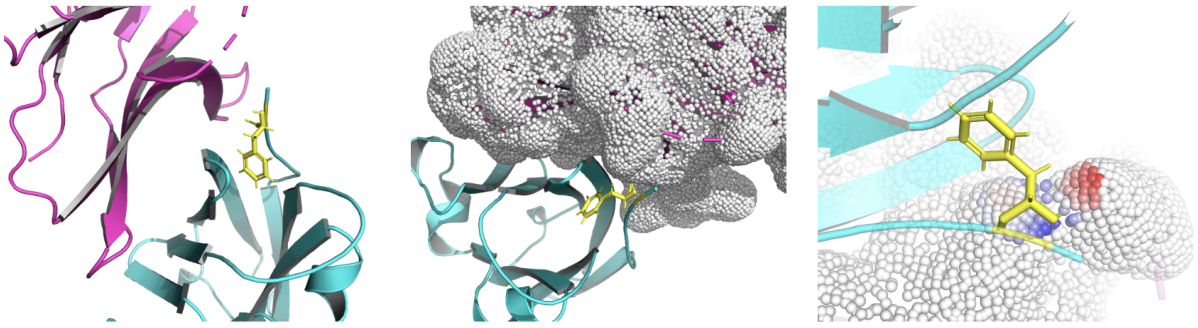


Figure 1: (a) An abstraction of a PPI. The cyan and magenta ribbons are two separate protein chains. A residue of the cyan protein is at the interface of these two proteins (yellow). Our algorithm predicts its identity using information from the magenta chain. (b) Surface point descriptions are generated based on the atoms of the magenta protein.
(c) A patch of the 50 closest surface points to the yellow residue are coloured based on their features.

## II. Models and Methods

### A. Method ouline

RosettaSurf[1], MaSIF and dMaSIF[5] have all been trained and evaluated on similar lists of known PPIs. We trained our model on many of these structures, but ensured that we did not use any residues from PDBs that are in the RosettaSurf test set. For our test set, we retrieved all residues used in the RosettaSurf publication's "single amino acid recovery"" section.

All of the PDB-files in the PPI-lists consisted of two chains or more chains, however, they were always split in two. Once we had a protein pair, residues were selected on one side and dMaSIF was used to calculate a surface description on the other [5].

This procedure is graphically depicted in Figure 1.

### B. Dataset Generation

The distribution of different residue types in the training dataset can be seen in Figure 2. The training set is inherently unbalanced. This is because of the nature of protein composition and folding, e.g. hydrophilic residues are more likely to be exposed.

The question might arise if the predicting algorithm would still be factually correct, especially considering the fact that the testing dataset (used in RosettaSurf [1]) is balanced. We solved this by adding weights to the loss functions. Using an unbalanced dataset with correctly adjusted weights should benefit prediction accuracy. That is simply because a much larger data is fed for the neural network to learn, which should increase the robustness of the predictions.

Furthermore, to perform a fair comparison, identical datasets are required. To achieve that, the subsequent step is to use $scripts/data_comparison_prep/get_specific_a.py$ to create the test and training set and compare the performance of Chopin software to the RosettaSurf [1]. The test set is

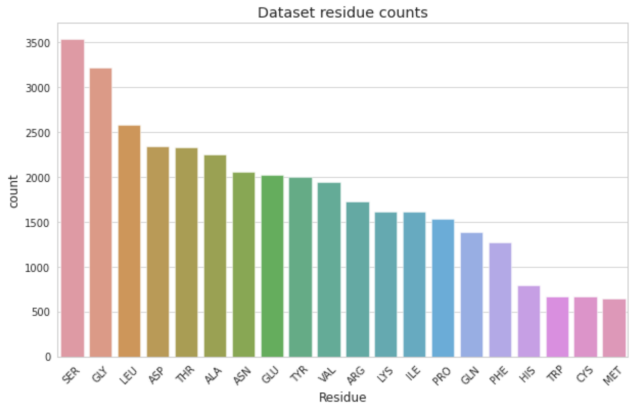


Figure 2: Distribution of residue abundance in the dataset

what we will perform predictions on is a test set of size of $200 * 20$ residues from 1495 PDB's. All other residues from those PDB's will serve as a training set.

### C. Feature generation

The surface data that was provided contained 34 features. The first 10 are geometric features of the structures, the next 6 are chemical features and the last 18 are learned features generated by dMaSIF. On top of this, we added three more features:

1) Distance from carbon atom in a residue to each of the points in a patch.
2) Angle for each residue between vector $C_\alpha$ to $C_\beta$ and $C_\beta$ to the surface point, calculated with the cross-product. This angle is called $\phi$ angle.
3) Angle of rotation between the nitrogen, $C_\alpha$ and $C_\beta$ and the surface point. This is the $\theta$ angle.

By adding these three features, the model can now learn about the spatial properties of the amino acids. The last two

angles are visualized in the appendix in figures 6 and 7.

### D. Data-set partitioning and training

In order to split the data, a dataset class, fit for dataloaders, was created. Afterwards, by using K-fold cross validation, the dataset was split in 4. Furthermore, by using subsamplers and dataloaders, the data could be iterated over the model in batch sizes of 1000 (train) and 200 (validation). This way, the big amount of data is split into smaller batches and the model does not get stuck on a too large dataset. Furthermore, a tensorboard was implemented to track the accuracies and losses in real time. If the tensorboard showed that the results were not at all as expected or showed signs of over-fitting, the model could be stopped prematurely.
After running the model for 200 train and test epochs, the best model (highest accuracy, lowest loss) was chosen and saved to perform the test set on.

### E. Determining the prediction's accuracy

The accuracy score $(TruePositives + TrueNegatives)/total$ is used to see which model can retrieve the most amino acids. This model is then chosen to take on the test set but eventually the receiver operating characteristic (ROC) curve is the used metric to evaluate the overall performance. The ROC curve plots the true positive rate (recall) against the false positive rate (fall-out). The ROC value is then calculated by taking the area under the curve. So, instead of taking only the true positives and true negatives, the ROC is a powerful tool because it takes into account all relevant quantities.

### F. Machine Learning models

The chosen neural network consists of 1 convolutional layer and 3 linear layers with 2 drop-out modules to counter overfitting. The exact order and values of the parameters can be seen in appendix figure 9.
The optimizer that is used for the model is the torch Adam optimizer. Furthermore, the loss function that has been chosen is the cross entropy loss. The problem can be seen as a multiclass classification problem and the cross entropy loss minimizes the distance between the probability distributions of the predicted and ground truth data, making it a suitable loss function a problem with multiple classes.

## III. Results and Discussion

In this section, results of the machine learning scheme as well as performance of the method in comparison to RosettaSurf are presented.

Figure 3 shows the learning curve of the convolution neural network for train and validation data sets. As it can be seen, the accuracy is increasing quite steeply for the first few tens of epochs and then settles for a more gradual increase at around 60 epochs. Both accuracies seem to be converging: for the validation set, the convergence accuracy is around 19 percent while for the train set it is 18 percent.

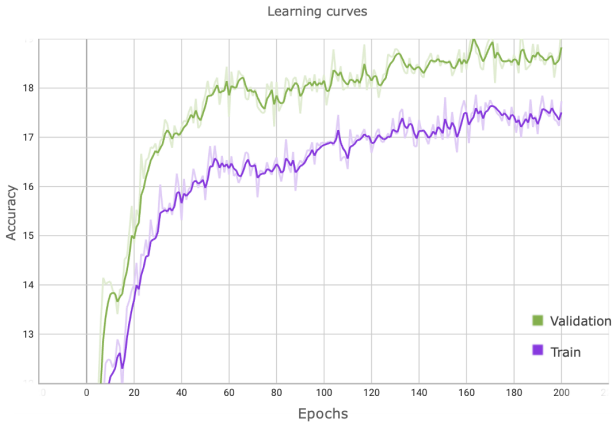


Figure 3: Learning curve for validation and training data sets.

Figure 5 presents the ROC curve per amino acid. As it can be seen, the accuracy of predictions varies for different amino acids. The neural network performs best for Glycine while its performance is the worst for Threonine.

Finally, Figure 4] presents the comparison of accuracy of predictions for the neural network created for the purposes of this project and the RosettaSurf algorithm [1]. As it can be seen, RosettaSurf vastly outperforms our neural network.

It shall be mentioned however, that this comparison is inherently unfair for the Neural Network algorithm created for the purposes of this report. Namely, when predicting the class of an amino acid at the specific location, RosettaSurf makes use of data about all the information (residues and atoms) from both sides of the protein-protein interaction when predicting type of the amino acid there. On the other hand, neural network of this project only has information about a few surface points on the other side of the protein-protein interaction. It is therefore unrealistic to expect results comparable to the RosettaSurf performance, especially at this stage of the project.

Furthermore, it shall be mentioned that some amino acids are similar to other ones in terms of their their electrostatic and shape characteristics and can thus be classified as similar groups. It can thus be concluded that while the algorithm not always predicts the right amino acid, it should be able to predict the right physical, chemical and electrostatic features of the amino acid that is at a given location. For example, Lysine and Arginine are very similar to one another. Thus, even if the neural network is unable to correctly distinguish between them, it is still able to determine their physical, chemical and geometrical correctly. That can be seen on Figure 8 in the Appendix.

## IV. Conclusion

From the results it can be seen that the accuracy of the Rosetta model is higher. One of the reasons for this is

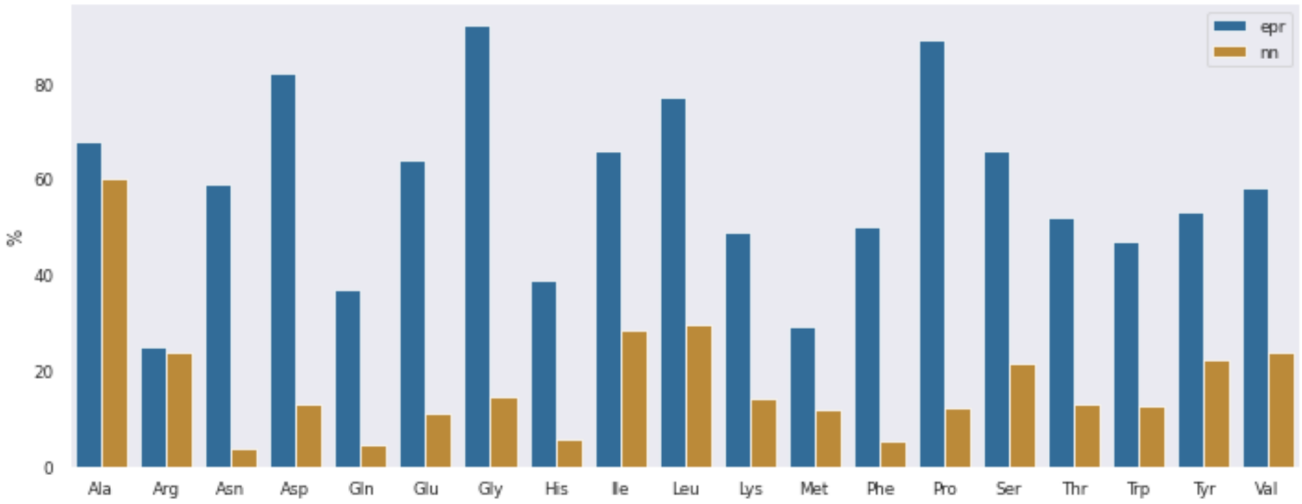


Figure 4: Comparison of the accuracy of predictions for the neural network algorithm and RosettaSurf.

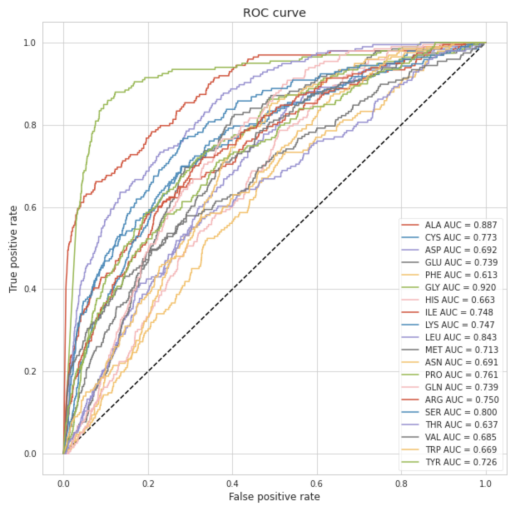


Figure 5: ROC curve for each aminoacid.

because Rosetta uses all the information from the atoms on both sides of the PPI. In contrast to our model where we try to predict the residue by looking at the surface of the opposing protein. However, judging from the heatmap in figure 8, it can be seen that many of the incorrect predictions were still predicted to be amino acid types with similar chemical properties. The similarity between these amino acids can be further investigated. In conclusion, the surface of an opposing protein provides enough information for a CNN to be able to predict what amino acid would be most suitable at this location.

## V. Recommendations

One of the main aspects of this project that should be researched further is different ways of expanding the dataset, either by increasing the number of features or by data augmentation. The aim of data augmentation is to increase the size of the dataset and thus improve the robustness of the training algorithm. Our idea for data augmentation is called ”multiple patches generation”.

In the original model, the surface patches only consist of the 50 closest surface points to the $C_\alpha$ atom. Therefore, the available training data is very limited. One way to solve this is by taking into account multiple patches of surface points, the available surface points increase largely. These patches are generated by taking the center of mass of the first patch and selecting N surface points. The N surface points each act as new initial points (previously the $C_\alpha$ atom) to which the 50 closest surface points form a patch around. Which N surface points and how many we take can be investigated in a further research. Another recommendation is to investigate whether the amino acids with similar properties can be used to replace one another in the PPIs.

## References

[1] A. Scheck, S. Rosset, M. Defferrard, A. Loukas, J. Bonet, P. Vandergheynst, and B. E. Correia, “RosettaSurf - a surface-centric computational design approach,” 2021.

[2] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis, “Highly accurate protein structure prediction with AlphaFold,” *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[3] T. Kortemme, D. E. Kim, and D. Baker, "Computational alanine scanning of protein-protein interfaces," *Sci. STKE*, vol. 2004, no. 219, p. l2, 2004.

[4] P. Gainza, F. Sverrisson, F. Monti, E. Rodola, M. M. Bronstein, and B. E. Correia, "MaSIF - deciphering interaction fingerprints from protein molecular surfaces," 2019.

[5] F. Sverrisson, J. Feydy, B. E. Correia, and M. M. Bronstein, "Fast end-to-end learning on protein surfaces," 2020.

## VI. Appendix

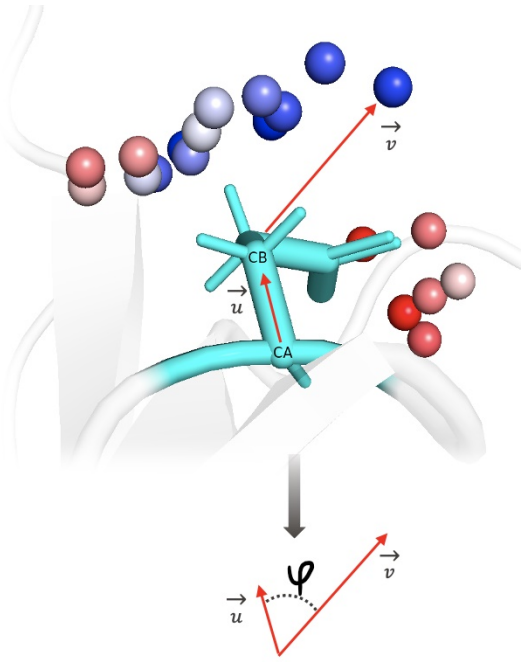


Figure 6: Phi angle

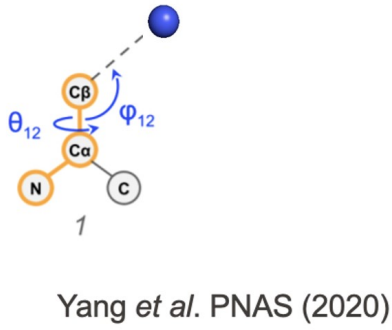


Figure 7: theta angle

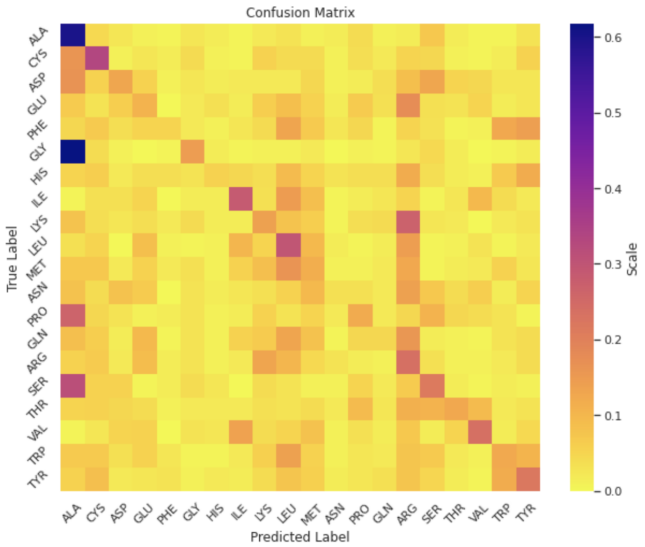


Figure 8: Heatmap

```python
class Model(nn.Module):
    def __init__(self, input_dim):
        super(Model, self).__init__()
        self.layer1 = nn.Conv1d(input_dim, 50, 1, stride=1)
        self.drop_layer_1 = nn.Dropout(p=0.5)
        self.layer2 = nn.Linear(50, 10)
        self.drop_layer_2 = nn.Dropout(p=0.8)
        self.layer3 = nn.Linear(500, 100)
        self.layer4 = nn.Linear(100, 20)


    def forward(self, x):
        x = torch.transpose(x, 1,2)
        x = F.relu(self.layer1(x))
        x = self.drop_layer_1(x)
        x = F.relu(self.layer2(x))
        x = x.view(-1,10*50)
        x = self.drop_layer_2(x) # dropout module
        x = F.relu(self.layer3(x))
        x = F.sigmoid(self.layer4(x))

        return x
```

Figure 9: Neural Network Architecture