Cracking the black box of deep sequence-based protein-protein interaction prediction

Judith Bernett[1], David B. Blumenthal[2,†], and Markus List[1,*†]

[1]Chair of Experimental Bioinformatics, TUM School of Life Sciences, Technical University of Munich, Freising, Germany
[2]Biomedical Network Science Lab, Department Artificial Intelligence in Biomedical Engineering, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany
[*]Corresponding author: markus.list@tum.de
[†]These authors contributed equally.

Abstract

Identifying protein-protein interactions (PPIs) is crucial for deciphering biological pathways and their dysregulation. Numerous prediction methods have been developed as a cheap alternative to biological experiments, reporting phenomenal accuracy estimates. While most methods rely exclusively on sequence information, PPIs occur in 3D space. As predicting protein structure from sequence is an infamously complex problem, the almost perfect reported performances for PPI prediction seem dubious. We systematically investigated how much reproducible deep learning models depend on data leakage, sequence similarities, and node degree information and compared them to basic machine learning models. We found that overlaps between training and test sets resulting from random splitting lead to strongly overestimated performances, giving a false impression of the field. In this setting, models learn solely from sequence similarities and node degrees. When data leakage is avoided by minimizing sequence similarities between training and test, performances become random, leaving this research field wide open.

Introduction

Proteins carry out essential biological functions, many of which require proteins to act jointly or to form complexes. Hence, identifying all pairwise interactions of proteins is an essential systems biology challenge towards understanding biological pathways and their dysregulation in diseases. Several technologies (e.g., yeast-2-hybrid screens, affinity purification mass-spectrometry) have been developed to unravel individual PPIs, yielding large-scale PPI networks[1]. As it is not feasible to study all protein pairs exhaustively, a plethora of computational methods have been developed to predict PPIs as a binary classification task. Such methods often use only sequence information in various machine learning (ML) strategies, ranging from classical support vector machines to the most complex deep learning (DL) architectures currently conceivable[2–19]. These DL methods typically report phenomenal prediction accuracies in the range of 95%-99%.

Though it was criticized that only a few of these methods have source code available and are reproducible[2,20], it has not yet been examined systematically whether and how such results are possible. Since proteins interact in 3D space, predicting an interaction should implicitly consider the 3D structure of complexes, binding pockets, domains, surface residues, and binding affinities. However, predicting protein 3D structure from sequence is an infamously hard problem that was solved only recently by AlphaFold2, using a very complex model architecture and vast resources[21]. Moreover, predicting the structure of

multi-chain complexes observed in PPIs remains an open challenge[22]. In the light of this, the observed high accuracies for predicting PPIs from sequence information alone seem dubious.

Few studies exist that shed light on the phenomenal accuracies reported for deep sequenced-based PPI prediction approaches: Almost all PPI datasets that have been used for evaluating such approaches are randomly split into train and test sets using cross-validation. Park & Marcotte[23] showed that this causes an inflation of prediction performance due to training data leakage[23–25]. Upon random splitting, the same proteins occur both in the train and the test set, such that these sets are no longer independent[26]. For an extensive definition of data leakage, see[27,28]. To quantify the effect of data leakage, Park & Marcotte[23] proposed three classes C1 (both proteins in a test pair occur in training), C2 (only one protein in a test pair occurs in training), and C3 (no overlap between training and test). Prediction accuracies usually drop significantly between C1 and C2 as well as between C2 and C3. It has also been shown that when datasets contain sequences with high pairwise sequence similarities, models overfit and accuracies are overestimated, giving a wrong impression of the state of the field[20,24,25]. Hamp & Rost[24] therefore extended the requirements by demanding that, for C3, no test protein should be sequence-similar to a training protein (for C2 only one, for C1 both), and obtained similar results as Park & Marcotte.

In addition to positive examples, it is crucial to add sufficiently realistic negative examples to both train and test sets. These can be randomly sampled by choosing protein pairs not reported as PPIs in public databases. To avoid using false negatives, it is common practice to choose proteins that are not annotated to the same cellular compartment and are thus not expected to interact in a cell. However, Ben-Hur & Noble[29] have shown that this approach makes the learning problem considerably easier than it is in reality. Moreover, protein pairs are tested for interaction in an artificial system[30] and databases thus frequently include interactions of pairs annotated to different cellular locations.

In this work, we examine systematically three scenarios that might explain why sequence-based models correctly predict whether two proteins p_1 and p_2 interact:

- Explanation 1: The models can detect patterns in the sequences of p_1 and p_2 that are responsible for whether they can interact (e.g., matching binding sites, domains, motifs).
- Explanation 2: The models can detect patterns in the sequences of p_1 and p_2 that individually explain whether the protein interacts or not. Based on these individual tendencies, they predict the interaction (e.g., p_1 has a promiscuous domain commonly found in hub proteins, making an interaction of p_1 with other proteins such as p_2 likely).
- Explanation 3: The models merely check whether p_1 and p_2 are similar to protein sequences p_1' and p_2' from the training set and make a prediction based on the interactions of p_1' and p_2' (e.g., if p_1' interacts with p_2', p_1 probably interacts with p_2 as well).

While, for many methods, the authors (implicitly) assume that their method's excellent prediction performance can be attributed to Explanation 1, we hypothesize that Explanation 2 and Explanation 3 are the actual drivers of high prediction accuracy. To investigate Explanation 2, we randomized the positive input network in the training data (but not in the test data) via degree-preserving rewiring. Hence, each protein's node degree is preserved, but the edges are no longer biologically plausible. If promiscuous proteins represented by hub nodes indeed drive prediction performance, we would expect only a moderate drop in the test accuracy.

To investigate Explanation 3, we carried out a two-fold strategy: On the one hand, we compared deep sequenced-based PPI prediction approaches against SPRINT[31] — an algorithmic PPI prediction approach based on local sequence alignment — as well as against simple baseline ML models which, by design, have access only to sequence similarity information. If sequence similarity indeed is a main driver of prediction performance, we would expect our baselines to achieve similar performances as the state-of-the-art DL methods. On the other hand, we partitioned the proteome into two blocks such that inter-block sequence similarities are minimized and selected positive PPI examples for train and test sets from different blocks

of the partition. This ensures that sequence similarity patterns learned during training cannot be leveraged at test time. If Explanation 3 is valid, we would expect a significant drop in prediction performance.

We carried out our analyses for four deep sequence-based PPI prediction methods[2,4,13], which we trained and tested on the same six publicly available and commonly used datasets (two datasets with yeast PPIs[12,32], four datasets with human PPIs[2,11,33]). Our results show that the excellent accuracies reported in the literature can be fully explained by training data leakage. More specifically, if pairwise sequence similarities are minimized between disjoint training and test sets, performance is random, proving that sequence similarity and node degree are the only relevant features in current sequence-based PPI prediction methods. Finally, we generated a gold standard dataset to enable data-leakage-free validation of future PPI prediction methods.

Results

Overview

We reviewed the literature for PPI prediction methods and their underlying datasets (Supplemental Table S1). For all 26 methods we found, extraordinary prediction performances are reported. However, source code is available only for nine of them, emphasizing the reproducibility crisis in ML-based science[27]. Since our focus was to understand how sequence information contributes to DL-based PPI prediction, we selected methods that we managed to reproduce with reasonable effort and which rely exclusively on sequence information. This reduced the number of DL methods to Richoux-FC, Richoux-LSTM[2], DeepFE[13], and PIPR[4].

Additionally, we tested SPRINT[31], a fast algorithmic method that uses only sequence similarities of protein pairs to predict PPIs. We also included two baseline ML models (Random Forest, SVM) that used dimensionality-reduced (PCA, MDS, node2vec) sequence similarity vectors as input for each protein. These baseline methods allowed us to assess the benefit of DL and to test the hypothesis that sequence similarity alone is already sufficient to achieve good prediction performance. Pairwise sequence similarities were pre-computed by SIMAP2[34]. Although sequence similarities were the only input feature, we note that the methods could learn node degrees implicitly during training. Supplemental Figure S1 depicts a schematic overview of the methods' principles.

Table 1: Overview of test datasets. n denotes the overall number of samples in the datasets (after balancing and removal of duplicates), i.e., the number of PPIs plus the number of randomly sampled non-edges. n_{method} is number of methods found in our literature review that were tested on the respective datasets. MRA denotes the median reported accuracy of the n_{method} methods tested on the respective datasets.

Dataset	Organism	n	n_{method}	MRA (in %)
HUANG[35]	Human	6690	4	98.43
GUO[32]	Yeast	11 162	14	94.75
DU[12]	Yeast	34 512	5	92.50
PAN[33]	Human	62 962	5	96.82
RICHOUX-REGULAR[2]	Human	79 868	2	88.10
RICHOUX-STRICT[2]	Human	68 144	2	77.29

We decided to test the methods on popular yeast and human datasets and the two datasets by Richoux et al.[2] (see Table 1 for an overview). The latter two datasets were included because of their size and the unique generation of the strict test dataset, which was designed to be free from hub biases. The two datasets were created from a larger dataset consisting of PPIs annotated in Uniprot which we used later on

for the partitioning task and refer to as RICHOUX-UNIPROT. All datasets were cleaned from duplicates and balanced. The original datasets were split 80/20 into training/test except for RICHOUX-REGULAR and RICHOUX-STRICT, which were already split into training/validation/test. Since we only used default hyperparameters, we did not need a validation set and added the validation to training for these two datasets. We chose the random 80/20 split since most reviewed methods report the mean accuracy of five-fold cross-validation[4–8, 11, 18, 33, 36, 37] or a random hold-out test set[2, 3, 12, 13, 32, 38–41].

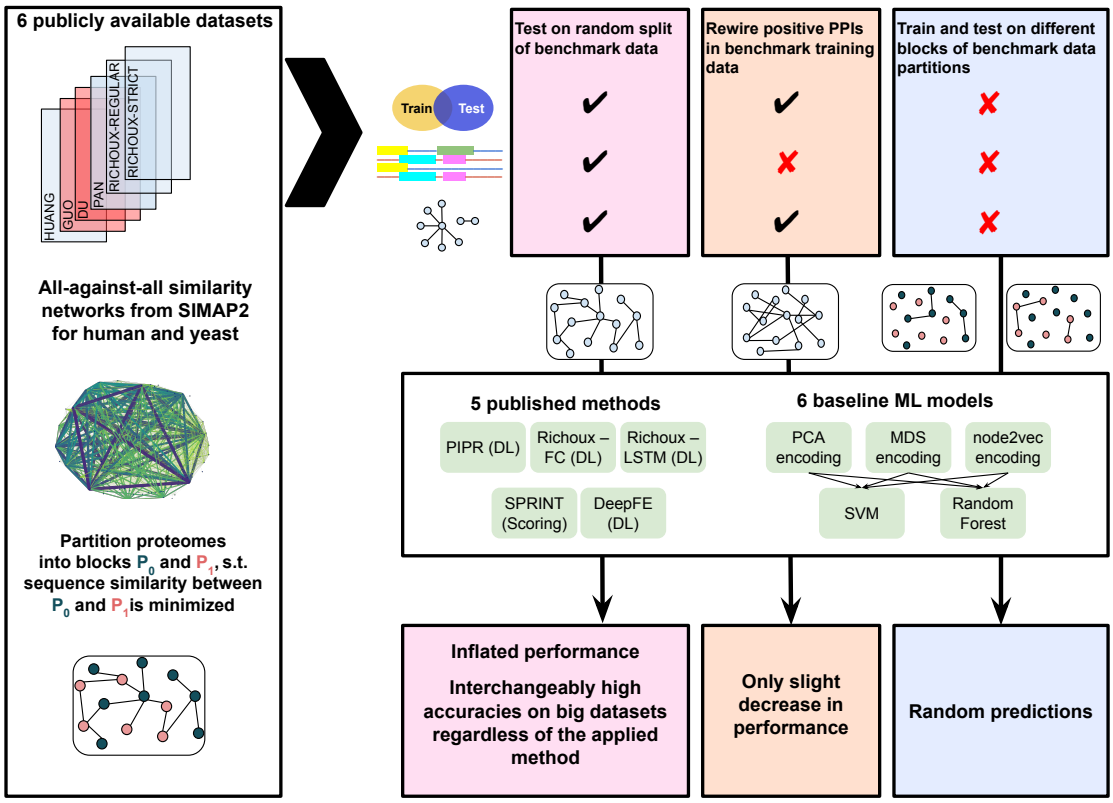


Figure 1: Overview of analyses. Six publicly available datasets were used for testing five published methods and six basic ML models that use sequence similarities as only input. We performed three tests to explain the phenomenal accuracies reported for DL methods: (1) We split the original data randomly into 80% train / 20% test, introducing data leakage through overlap of train and test proteins. Methods could learn from node degree biases and sequence similarities. This yielded inflated performance estimates and interchangeably high accuracies for DL and basic models on large enough datasets. (2) We rewired the positive train PPIs such that models could only learn from node degrees. Nevertheless, performance estimates only decreased slightly. (3) We partitioned the human and yeast proteomes into two blocks P_0 and P_1 such that proteins from different blocks have pairwise dissimilar sequences and assigned PPIs (p_1, p_2) to blocks $INTRA_0$, $INTRA_1$, and $INTER$, depending on whether p_1 and p_2 are both contained in P_0 or P_1, or fall into different blocks of the partition. When trained on $INTRA_0$ and tested on $INTRA_1$ (no overlap between train and test data, models could neither learn from sequence similarity nor from node degrees), all tested models predicted PPIs randomly.

Figure 1 provides an overview of our analyses. We first consider a random split into train and test set, which we expect to introduce data leakage (see Methods for details). To test how much the models learn

from node degree only (Explanation 2), we next rewired the positive PPIs (edges in PPI networks) in all training folds. For this, we randomly re-assigned all edges but preserved the expected node degree for each protein, rendering the new positive PPI networks biologically meaningless (see Figure 6). Finally, we used the KaHIP[42] method with pre-computed SIMAP2[34] bitscores as input to partition the human and yeast proteomes into two blocks, P_0 and P_1, such that pairs of protein sequences from different blocks are dissimilar. Then, for each dataset, all PPIs (p_1, p_2) were assigned to three blocks $INTRA_0$, $INTRA_1$, and *INTER*, depending on whether p_1 and p_2 are both contained in P_0, whether they are both contained in P_1, or whether they are contained in different blocks of the partition $\{P_0, P_1\}$. If Explanation 3 applies, we expect a significant drop in accuracy if we train on the PPIs contained in $INTRA_0$ and test on the ones contained in $INTRA_1$, since there is no direct data leakage ($P_0 \cap P_1 = \emptyset$) and a minimized amount of indirect data leakage due to sequence similarity. If we train on *INTER* and test on $INTRA_0$ or $INTRA_1$, we expect a smaller drop in accuracy, since there is data leakage.

Results on randomly split original benchmark datasets

Figure 2 shows the results we obtained when randomly splitting the original benchmark datasets into 80% train / 20% test. Since all published methods were reported to show close to perfect performances, we expected roughly equal results across methods within each dataset. As larger data sets are more prone to data leakage, we further expected accuracy to increase with dataset size for random splitting.

Comparing the results for all datasets except for the RICHOUX-STRICT dataset, we can see that the Area under the Curve (AUC) values of SPRINT rise with the number of unique proteins in the dataset (Supplemental Table S2). SPRINT performs equally well as the DL models on large datasets, which shows that finding similar subsequences to predict PPIs is already sufficient to reach excellent accuracy when proteins between the train and test set are shared.

For almost all methods, performances are exceptionally high on the HUANG and PAN datasets. This can be explained by looking at the node degree distributions of the positive and negative datasets (Figure 2b).While for the other datasets, both distributions follow the power law (Supplemental Figure S3), the negative examples were sampled uniformly for the HUANG and PAN datasets. Methods can thus mostly distinguish between positive and negative examples by node degree alone.

Except for SVM-PCA and SVM-node2vec, the performance of the baseline ML methods is virtually interchangeable and roughly equal to the DL methods on the larger datasets. We cannot offer a reasonable explanation for the worse performance of SVM-PCA and SVM-node2vec. Considering that the SVM generally performs worse than the Random Forest, we can hypothesize that the more complex MDS encoding somehow mitigates this.

As expected, the performance of all methods drops significantly on RICHOUX-STRICT, on which SPRINT performs best. As shown in Figure 2c, all datasets except for RICHOUX-STRICT include the vast majority of the proteins in both train and test set. Consequently, RICHOUX-STRICT is less prone to training data leakage, explaining the observed results. In the presence of data leakage, robust predictions can be made based on node degree and sequence similarity even with basic ML models. For RICHOUX-STRICT, the overlap is still almost 50%, but it is free from hub biases. SPRINT does not use node degrees for its predictions, so the number of protein pairs seen in training seems to be large enough for SPRINT to find similar subsequences for the RICHOUX-STRICT test set. The drop in accuracy of about 20% represents the performance of the other methods when node degrees cannot be used anymore and when direct data leakage is approximately halved (Supplemental Table S2).

For all benchmark datasets, the overall number of proteins occurring in the datasets is far from the real number of proteins (Supplemental Table S2). It can hence be expected that the models overfit extremely on a specific subset and will not generalize when presented with unknown proteins.

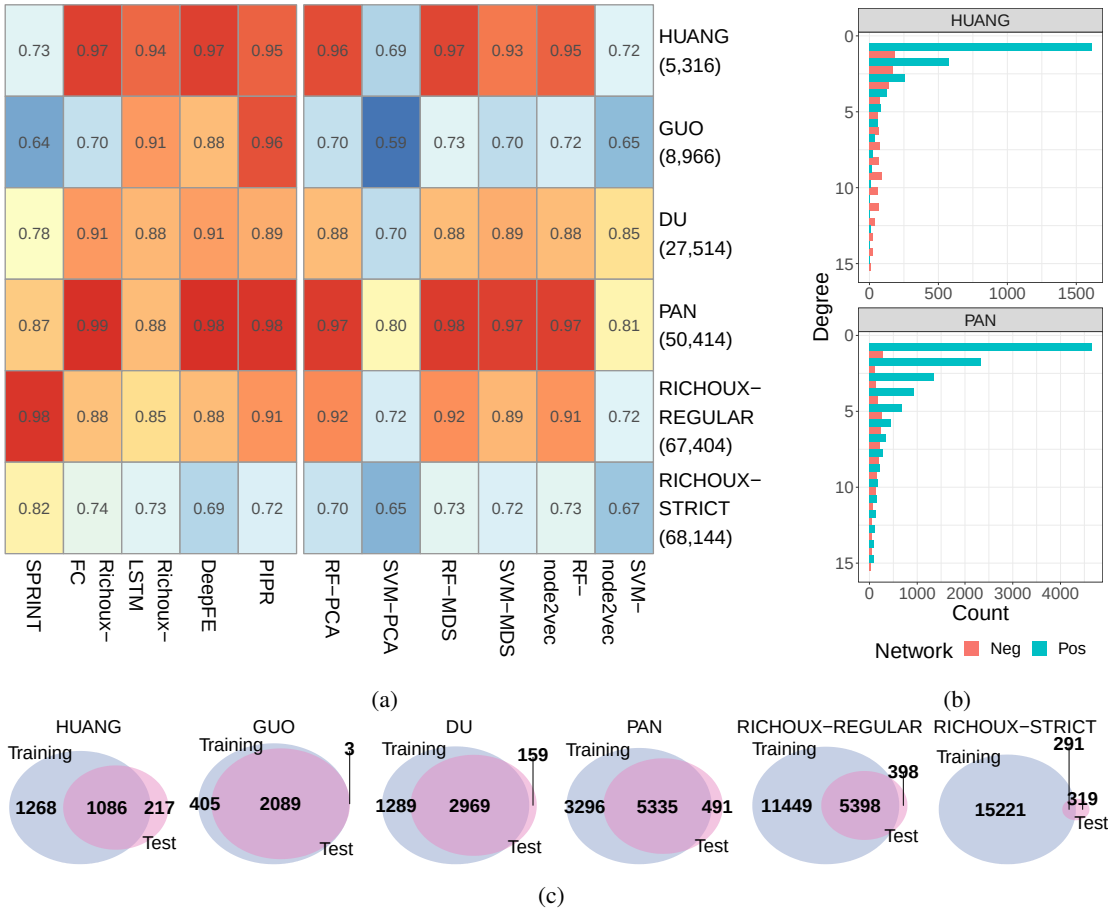
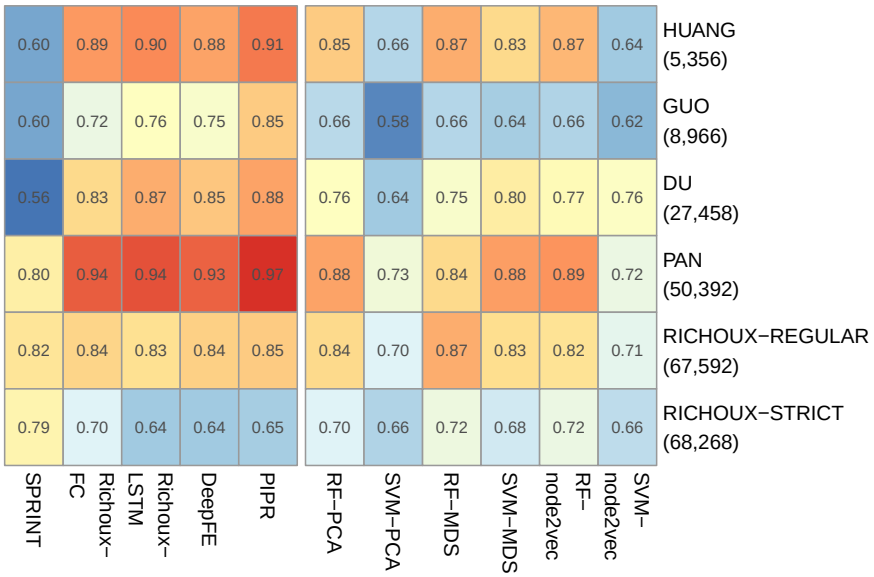


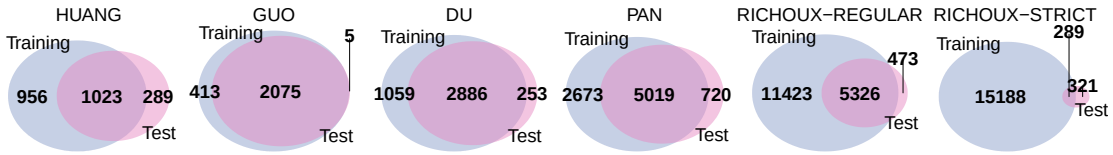
Figure 2: Results obtained on randomly split original benchmark datasets. **(a)** AUCs on test sets for SPRINT, accuracies for all other methods. Y-axis labels correspond to the dataset, the numbers of samples in the training data are shown in parantheses. AUC values of SPRINT rise with the number of proteins in the dataset. Performances are exceptionally high on the HUANG and PAN dataset. For four out of six baseline ML models, performances are comparable to the performances of the DL models. All performances drop significantly on RICHOUX-STRICT. **(b)** Node degree distributions of the positive and negative PPI datasets for HUANG and PAN. Only the node degrees of the positive PPIs follow a power law distribution. **(c)** Overlap of proteins occurring in the original training and test sets.

Rewiring tests

We investigated how node degree-preserving rewiring of the positive training set affects the methods in general (Explanation 2). The results are shown in Figure 3a. As edges are no longer biologically meaningful, the methods can only learn from node degrees to make correct predictions in the unperturbed test sets. If protein p is a hub protein in the training data, any PPI involving p in the test data is likely to be predicted as positive. Sequence similarities can only help when p' is similar to the hub protein p. Then, p' is more likely to be a hub protein (e.g., because of a shared promiscuous domain). However, sequence similarities hinting at binding sites or interacting domains cannot be used for prediction because of the rewired training data. While a slight drop in accuracy compared to the original performance would support



(a)



(b)

Figure 3: Results obtained on datasets with randomly rewired positive PPIs in the train sets. **(a)** AUC values on unmodified test sets for SPRINT, accuracies for all other methods. Y-axis labels correspond to the dataset, the numbers of samples in the training data are shown in parantheses. Performances fell only slightly compared to the results on the original datasets. Very high accuracies can still be reached for HUANG and PAN. For larger datasets, basic ML models perform approximately as well as the DL models. SPRINT performs almost randomly on small datasets but still reaches accuracies around 80% on the PAN and RICHOUX datasets. **(b)** Overlap of proteins occurring in the rewired training and test sets.

the validity of Explanation 2, a significant drop would indicate that the models do not learn from node degrees alone. The extent of data leakage remains comparable to the original datasets (Figure 3b).

Indeed, the performances fell slightly compared to the results on the original datasets for all methods. Very high accuracies can still be reached on the datasets HUANG and PAN. This is in accordance with our observations from Figure 2b and our findings on the original datasets. For these two datasets, the node degree distribution of the positive PPIs (power law) is not equal to the one of the negative PPIs (uniform). Therefore, the methods learn that nodes with few edges mostly occur in the positive dataset and can fully leverage this information to predict the unrewired test sets.

Again, the baseline models SVM-PCA and SVM-node2vec perform worse than the other baseline models (see previous section). Generally, the baseline models are outperformed slightly by DL models, except for RICHOUX-REGULAR and RICHOUX-STRICT, where the performance is similar. It is possible that the DL methods are better at recognizing node degree biases compared to basic models which need larger datasets to achieve this.

SPRINT shows comparably poor performance on the smaller datasets but achieves an AUC of up to 82% on large datasets. We cannot fully explain these high AUC values. SPRINT searches for sequence similarities in potentially interacting protein pairs and does thus not benefit from node degree information. While we see a significant drop in the magnitude of the scores compared to the original datasets (Supplemental Figure S5), scores of interacting proteins are still higher than those of non-interacting proteins.

Overall, we can confirm that the methods are biased by node degree information: Accuracies up to 97% can still be reached in spite of rewiring the training data.

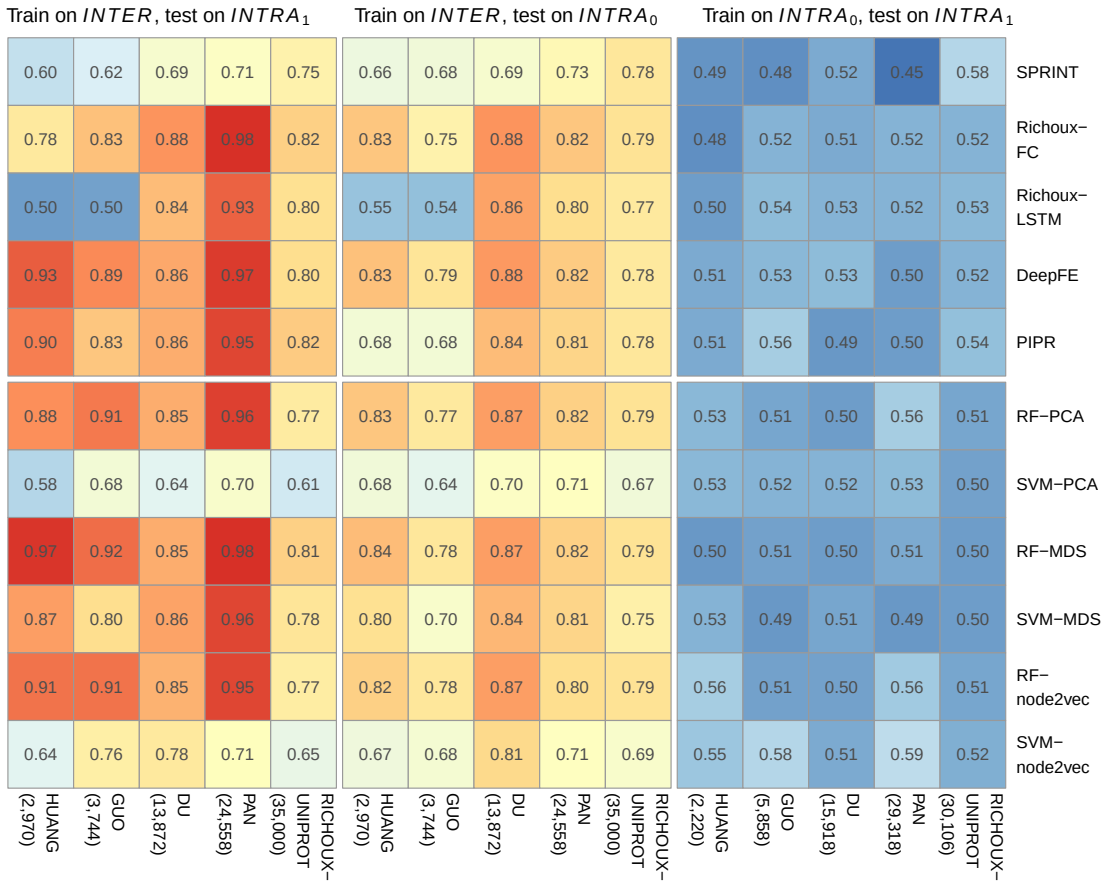
Partitioning tests

Running the baseline methods on the original datasets was a positive test for Explanation 3, as we expected similar performance for DL and basic ML models. Indeed, our results confirm this expectation (see Figure 2a). The partitioning tests served as a negative test for Explanation 3. If the explanation were valid, we would expect the performances to drop significantly when the models are trained on $INTRA_0$ and tested on $INTRA_1$. We expected this for both the DL and baseline ML models.

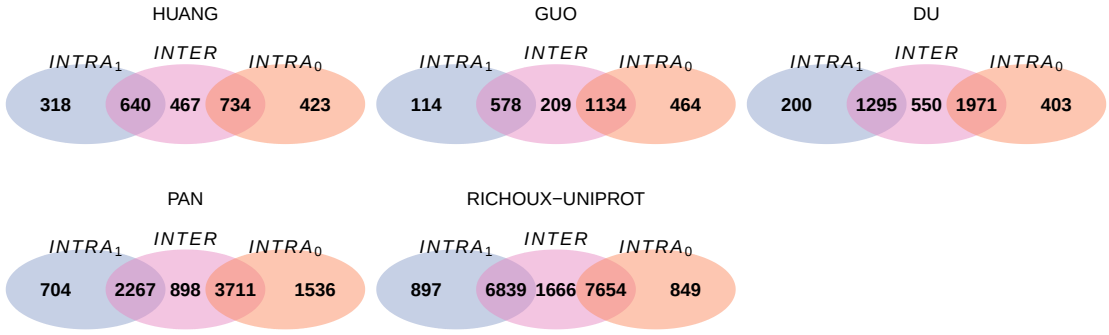
The results of the partitioning tests are shown in Figure 4. Notably, all training dataset sizes were approximately halved because of the partitioning strategy. While *INTER* and $INTRA_0$ are approximately equal in size, $INTRA_1$ is considerably smaller (see Methods for details). Additionally, when training on *INTER*, the train and test set overlaps constitute a larger factions of the test sets for when testing on $INTRA_1$ than when testing on $INTRA_0$ (Figure 4b).

Indeed, we observed random performances for all methods trained on $INTRA_0$ and tested on $INTRA_1$. The results show that, when the test sets do not suffer from data leakage, the methods do not learn any higher-level features during training that they can apply to unseen data. Instead, models overfit on the interaction patterns of the training proteins. Predictions become random when the test set does not contain these proteins (or highly similar proteins). For RICHOUX-UNIPROT, almost all proteins are predicted as non-interacting (specificity ≈ 1.0, see Supplemental Figure S9).

Overall, the performances obtained after training on *INTER* were excellent, especially when the methods were tested on $INTRA_1$. Only Richoux-LSTM yielded random predictions for the small datasets due to overfitting, which was foreseeable when running 100 epochs on less than 4000 data points. That the performances on $INTRA_1$ were better than on $INTRA_0$ can be explained by examining Figure 4b. For all but the RICHOUX-UNIPROT blocks, the overlap between *INTER* and $INTRA_1$ is higher, resulting in more data leakage. The differences between the overlaps are the highest for GUO and PAN, for which we can also see the largest drops in performance. When training on the *INTER* block of the RICHOUX-UNIPROT dataset, the amount of data leakage (difference: 1.6%) and the results are roughly equal when testing on the $INTRA_0$ block and when testing on the $INTRA_1$ block (mean $\pm$ standard deviation of differences: 0 ± 0.04). This shows that the effect of the resulting data leakage is considerable even when only one of the proteins of each training PPI occurs in the test set.



(a)



(b)

Figure 4: Results of partitioning tests. **(a)** AUC values for SPRINT, accuracies for all other methods. X-axis labels correspond to the dataset, the numbers of samples in the training data are shown in parentheses. Performances drop to random for all methods when trained on $INTRA_0$ and tested on $INTRA_1$. Performances obtained from training on *INTER* are still excellent, especially when tested on $INTRA_1$. Generally, training on *INTER* and testing on $INTRA_1$ resulted in higher accuracies than training on *INTER* and testing on $INTRA_0$. **(b)** Overlap of proteins occurring in the different blocks of the partitions of the benchmark datasets.

Runtime

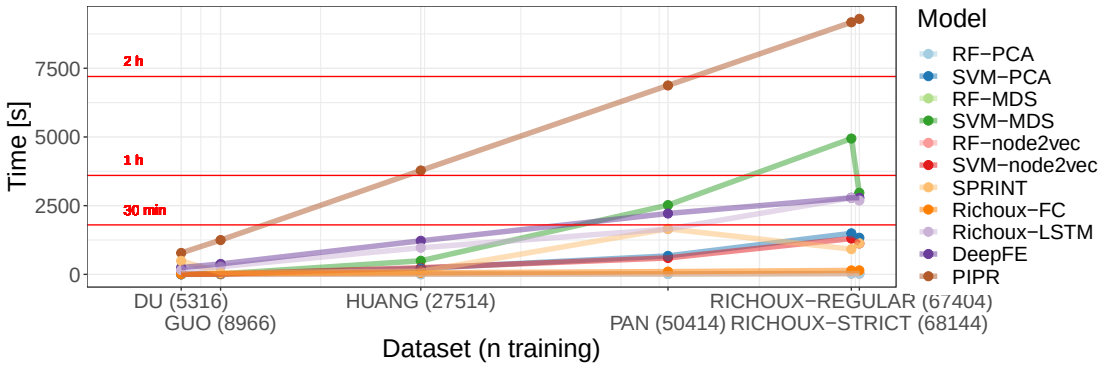


Figure 5: Overall runtime for training and testing on the original datasets. Most runtimes increase linearly with size of the training dataset. PIPR has by far the highest runtime while Richoux-FC has the lowest. The SVM-based baseline methods converge slower than the RF-based ones.

Runtime typically grows linearly with the size of the training dataset (Figure 5). An exception is SPRINT which always loads the preprocessed proteome before reading in the positive training PPIs. As this data contains the computed similar sub-sequences, the file is much more extensive for the human proteome compared to yeast; therefore, SPRINT takes longer on human compared to yeast datasets. PIPR (50 epochs) has by far the highest runtime. Regarding our baseline ML models, Random Forest is much faster than SVM for large datasets. This observation was also made by Soleymani et al.[43], who criticized that prediction methods using LSTMs and GRUs are computationally demanding and converge slowly and that SVM-based methods are unsuitable for large datasets[43]. The fastest method is Richoux-FC, which only runs for 25 epochs instead of 100, like Richoux-LSTM.

Combined with the results from the previous sections, it is evident that an enormous amount of resources can be saved by using simple methods to predict PPIs like scoring algorithms, Random Forests, or multilayer perceptrons with few layers. These methods achieve similar results as the costly deep neural networks and can confidently predict PPIs using sequence similarities and learned interaction patterns for known proteins. When no sequence similarity information is available and PPIs should be predicted for proteins unseen in the training data, all sequence-based methods fail such that there is again no advantage in using approaches based on DL.

Gold standard dataset

We showed that existing DL models fail to extract more complex sequence features for predicting PPIs. For the design of more advanced machine learning strategies, we provide a leakage-free human gold standard data set for training, validation, and testing (available at `https://doi.org/10.6084/m9.figshare.21591618.v1`)[44]. The positive dataset was created using data from HIPPIEv2.3[45]. Negative PPIs were sampled randomly, but such that individual node degrees are preserved in expectation (Supplemental Figure S3). This dataset was split using our partitioning strategy with KaHIP, i.e., there are no overlaps between the three sets and a sequence similarity is minimal. Both training and validation datasets are large enough to allow DL methods to avoid overfitting. Additionally, the sets are redundancy-reduced w.r.t. pairwise sequence similarity using CD-HIT at a 40% threshold[46]. As a result, proteins are also pairwise dissimilar within their set, such that models have to extract features beyond sequence similarity to achieve good performance.

To confirm that our gold standard data set shows the expected behavior, we evaluated all methods on it. Because SPRINT and our baseline models do not have any tunable parameters, we collapsed the training and validation set for their training. All methods were evaluated on the test set. Indeed, performances were random for all methods (Supplemental Table S5). None of the methods could extract any higher-level features during training that could be applied to predict the test set.

Discussion

We have conclusively shown that the problem of binary PPI prediction is not solved but wide open. Numerous publications report accuracy values between 90% and 100% and fuel a feedback loop of over-optimism. We have shown that their prediction estimates can be solely attributed to data leakage caused by random splitting into train and test sets. The datasets used in literature cause the models to overfit based on protein homology and node degree information. More complex sequence features representing binding pockets, protein domains, or similar motifs are not extracted. Instead, methods depend on global sequence similarity and node degree.

We have reached this conclusion using three experimental settings: Firstly, we have shown that after random 80/20 splitting of the datasets, DL and baseline ML methods yield interchangeably high results on all datasets. SPRINT, the most straightforward method, performs as well as the DL models on large datasets, which shows that finding similar subsequences to predict PPIs is already sufficient to reach excellent accuracy. When the methods cannot use information about hub proteins (RICHOUX-STRICT test set), performance drops for all methods.

Secondly, we have demonstrated that biologically meaningless edges which preserve the expected node degree do not lead to random predictions. While, for all methods, performance measures fall compared to the original datasets, accuracies up to 97% can still be reached for the DL methods and 89% for the baseline methods. Hence, the models can confidently predict PPIs from node degree only.

Finally, we proved that excluding training proteins from the test set and minimizing pairwise sequence similarities between training and test sets strips all methods of their predictive power. Taken together, our results show that DL methods do not learn any higher-level structural features. Conversely, we observe strongly elevated performance scores after training the methods on a data set with shared proteins.

In this study, we focused exclusively on sequence-based methods. In future work, it would be interesting to see if our findings translate to methods predicting PPIs from 3D structures. Since proteins interact in a folded state, methods using this information might extract the actual underlying patterns and find matching sites and domains.

We would also like to point out that PPI networks neglect differences in the interactions of protein isoforms[47], where, for instance, the absence of a binding domain will limit a protein's set of interaction partners. Moreover, most proteins work in larger complexes, i.e., they form long-lasting interactions between two or more proteins. These complexes might then perform their functions by interacting transiently with other proteins or protein complexes. If we want to predict and understand the underlying mechanisms of PPIs, we have to consider non-binary interactions as well as the difference between transient interactions and protein complex formation.

Binary PPI predictions are already very accurate for proteins seen during training or for proteins that share similar (sub-)sequences. We have seen that high accuracies can be obtained with simple scoring methods or fast-converging ML models. We appeal to the community to first try simple methods that cost fewer resources before moving on to complex and deep model architectures. Not only are these models prone to overfitting, but they also waste an unnecessary amount of time, memory, energy, and CO2.

Nevertheless, as impressively shown by AlphaFold2, DL models have tremendous potential. We speculate that extensive training data leakage has concealed the true challenge of binary PPI prediction. For future ML efforts, we thus provide a large gold standard training, validation, and test set that is free

from data leakage and has minimized pairwise sequence similarities. With this, we hope to kindle renewed interest in this ML challenge and to motivate further progress in the refinement of existing PPI prediction networks.

Methods

Datasets

We tested on the six datasets summarized in Table 1. The yeast dataset GUO[32] contains 5594 positive PPIs from DIP with less than 40% pairwise sequence identity and 5594 negative PPIs generated from pairs of proteins appearing in the positive set which, according to Swiss-Prot annotations, are expressed in different subcellular locations. The yeast dataset DU[12] was generated similarly and contains 17257 positive and 48594 negative PPIs. The human dataset HUANG[35] contains 3899 positive experimentally verified PPIs from HPRD with less than 25% pairwise sequence identity and 4262 negative PPIs which were generated as for the datasets GUO and DU. The human dataset PAN[33] contains 36630 positive PPIs from HPRD and 36480 negative PPIs, which where generated by combining protein pairs obtained via the approach described above with non-interacting pairs contained in the Negatome[48]. The human dataset RICHOUX-REGULAR contains positive PPIs retrieved from UniProt and negative PPIs which were generated by randomly pairing proteins from the positive set. Sequences were filtered to be at most 1166 amino acids long, mirror copies were added (for each PPI (p_1, p_2), add (p_2, p_1)), and the resulting dataset was split into a training ($n_{\text{train}} = 85104$), a validation ($n_{\text{val}} = 12822$), and test fold ($n_{\text{test}} = 12822$). The human dataset RICHOUX-STRICT[2] was constructed from the RICHOUX-UNIPROT dataset as follows: PPIs whose involved proteins appear less than 3 times were assigned to the test fold. The remainder was redistributed among the training and validation datasets. The resulting sizes of the training, validation, and test folds are, respectively, $n_{\text{train}} = 91036$, $n_{\text{val}} = 12.506$, and $n_{\text{test}} = 720$.

The six datasets were cleaned from duplicates and checked for overlaps. The training and validation folds in RICHOUX-REGULAR and RICHOUX-STRICT were joined for all analyses. All other datasets were randomly split into a train (80%) and test (20%) set. A validation set was not needed since we omitted hyperparameter optimization. After splitting, the datasets were balanced either by randomly dropping negatives or by sampling new negatives such that both proteins are already part of the dataset and that the interaction is not part of the existing positive or negative interactions (Table 2).

Rewiring tests

In order to test how much the models learn from node degree only, we rewired the positive PPIs of all described training datasets such that all proteins keep their original degree in expectation. However, the edges are newly assigned, rendering them biologically meaningless. For this, we used the `expected_degree_graph()` function of the NetworkX 2.8 Python package. Given the list $(w_0, w_1, \ldots, w_{m-1})$ of node degrees in the original network (positive PPIs in training fold), the function constructs a graph with m nodes and assigns an edge between node u and node v with probability $p_{uv} = \frac{w_u w_v}{\sum_k w_k}$. Again, all datasets were checked for duplicates and overlaps and were balanced after splitting, resulting in the counts summarized in Table 3.

A significant drop in accuracy compared to the performance on the original dataset could indicate that the models learn from the sequence features. However, a small drop would indicate that the models mostly memorize node degrees and assign their predictions based on whether or not the protein is overall likely to interact (Explanation 2).

Table 2: State of the benchmark datasets after cleaning and balancing: n denotes the overall number of samples in the datasets, i.e., the number of PPIs plus the number of randomly sampled non-edges. n_{train} and n_{test} are defined analogously for the train and test sets. The modifications were done to clean and balance the original benchmark datasets, i.e., to ensure that the number of positive PPIs (edges) equals the number of negative PPIs (non-edges) in the train and test splits of all datasets.

Dataset	n	n_{train}	n_{test}	Modifications
GUO	11 162	8966	2196	24 duplicates, sampled 35 negatives for training, dropped 37 negatives for testing
DU	34 512	27 514	6998	2511 duplicates, dropped 23 157 negatives for training and 5680 for testing
HUANG	6690	5316	1374	0 duplicates, dropped 619 negatives for training and 110 for testing
PAN	62 962	50 414	12 548	55 duplicates, dropped 1678 negatives for training and 476 for testing
RICHOUX-REGULAR	79 868	67 404	12 464	5047 duplicates, dropped 25 475 negatives for training and 342 for testing
RICHOUX-STRICT	68 664	68 144	520	5341 duplicates, dropped 30 057 negatives for training and 200 for testing

Partitioning tests

To explore Explanation 3 which hypothesizes that the models mostly learn from sequence similarities (see Introduction), we partitioned the yeast and human proteomes into two disjoint subsets P_0 and P_1 such that proteins from different subsets are pairwise unsimilar. For this, we first exported the yeast and human similarity networks by SIMAP2 as METIS files with bitscore weights. This resulted in weighted similarity networks with, respectively, 6718 nodes and 92 409 edges (for the yeast proteome) and 20 353 nodes and 1 900 490 edges (for the human proteome). In the similarity networks, bitscore edge-weights increase with increasing pairwise sequence similarity.

These similarity networks were then given to the KaHIP KaFFPa algorithm (desired output partitions: 2, pre-configuration: strong), which (heuristically) solves the following problem: Given a graph $G = (V, E, \omega)$ with non-negative edge weights $\omega : E \to \mathbb{R}_{\geq 0}$, it partitones V into blocks P_0 and P_1 such that, for all $i \in \{0, 1\}$, it holds that $|P_i| \leq (1+\varepsilon)\lceil \frac{|V|}{2} \rceil$ (partition is almost balanced) and the total cut size $\omega(P_0, P_1) = \sum_{u \in P_0} \sum_{v \in P_1} \omega(uv) \cdot [uv \in E]$ is minimized (the hyperparameter ε was left at the default $\varepsilon = 0.03$). For both the yeast and the human proteome, we hence obtained two disjoint subsets of proteins such that the overall pairwise sequence similarity between the subsets (sum of bitscores along the cut) is minimized.

For the yeast proteome, this resulted in $|P_0| = 3458$ and $|P_1| = 3260$; for the human proteome, we obtained $|P_0| = 10481$ and $|P_1| = 9872$. Based on the partition $\{P_0, P_1\}$ of the human and yeast proteomes, we then partitioned each PPI dataset into three blocks INTRA_0, INTRA_1, and INTER. Each PPI (p_1, p_2) was assigned to one of these blocks as follows:

- We assigned (p_1, p_2) to the block INTRA_0 if $p_1, p_2 \in P_0$.
- We assigned (p_1, p_2) to the block INTRA_1 if $p_1, p_2 \in P_1$.
- We assigned (p_1, p_2) to the block INTER if $p_1 \in P_0 \wedge p_2 \in P_1$ or $p_1 \in P_1 \wedge p_2 \in P_0$.

Again, all datasets were cleaned from duplicates and balanced after partitioning. If additional negatives had to be sampled, they were sampled from the proteins of the respective block. This yielded the numbers

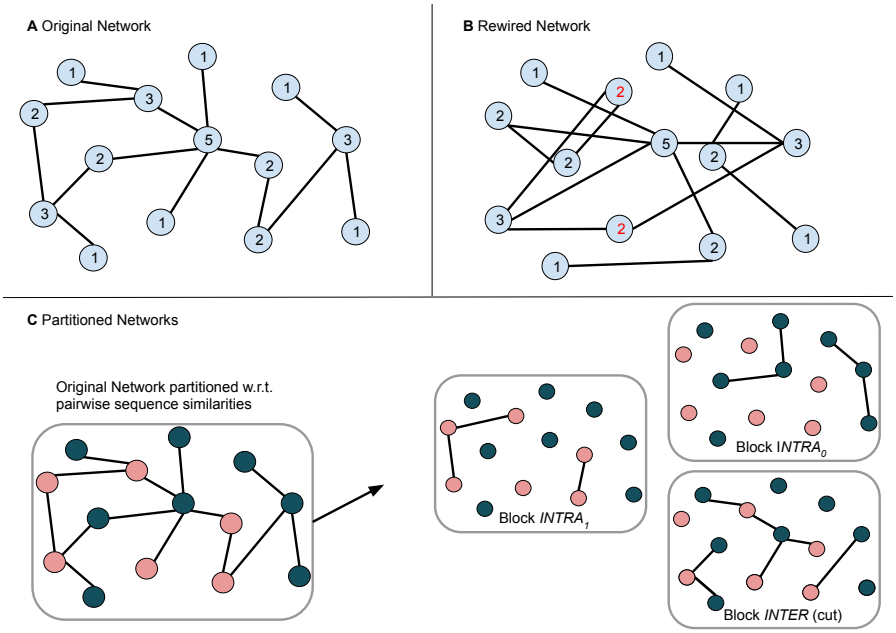


Figure 6: Concept of the rewiring and partitioning strategies. When the original network (A) is rewired, all node degrees stay the same in expectation but the edges are no longer meaningful (B). For the partitioning tests (C), the proteome is partitioned into two blocks (pink and green nodes) such that pairwise inter-block sequence similarity is minimized. Then, the PPIs from the original network (A) are partitioned based on which block the involved proteins are contained in ($INTRA_0$: both proteins contained in green block, $INTRA_1$: both proteins in contained pink block, *INTER*: proteins contained in different blocks).

of samples shown in Table 4. Methods were then either trained on block $INTRA_0$ and tested on block $INTRA_1$ or trained on block *INTER* and tested on the blocks $INTRA_0$ and $INTRA_1$. Following Explanation 3, we expected the most significant drop in accuracy compared to the original performance when training on $INTRA_0$ and testing $INTRA_1$. We expected a smaller drop in performance when training block *INTER* and testing on $INTRA_0$ and $INTRA_1$, since then, for approximately half of the test PPIs, sequence similarity information is available at test time. Note that, from the three datasets published by Richoux et al.[2], we partitioned the dataset RICHOUX-UNIPROT as it contains the largest number of unique proteins.

Construction of gold standard dataset

The whole human proteome was split into three parts by running KaHIP on the all-against-all sequence similarity matrix from SIMAP2. When configured to output a three-way partition, KaHIP partitions the node set V of an edge-weighted graph $G = (V, E, \omega)$ into blocks P_0, P_1, and P_2 such that the cut size

$$\omega(P_0, P_1, P_2) = \sum_{\{i,j\} \in \binom{\{0,1,2\}}{2}} \sum_{u \in P_i} \sum_{v \in P_j} \omega(uv) \cdot [uv \in E] \tag{1}$$

is minimized and $|P_i| \leq (1+\varepsilon)\left\lceil \frac{|V|}{3} \right\rceil$ holds for all $i \in \{0,1,2\}$. This resulted in 6987 proteins in P_0, 6987 proteins in in P_1, and 6379 proteins in P_2.

831 933 positive PPIs were downloaded from the HIPPIE database[45] (version 2.3). Mapping all 18 909 unique IDs to UniProt IDs using the UniProt mapping tool resulted in 17 269 unique proteins and 689 735

Table 3: State of the benchmark datasets after rewiring the positive training PPIs and balancing the datasets.

Dataset	n	n_{train}	n_{test}	Modifications
GUO	11 256	8966	2290	24 duplicates, dropped 12 negatives for training, sampled 57 negatives for testing
DU	34 416	27 458	6958	2511 duplicates, dropped 23 086 negatives for training and 5711 for testing
HUANG	6722	5356	1366	0 duplicates, dropped 595 negatives for training and 118 for testing
PAN	62 974	50 392	12 582	55 duplicates, dropped 1706 negatives for training and 442 for testing
RICHOUX-REGULAR	80 056	67 592	12 464	5047 duplicates, dropped 25 382 negatives for training and 342 for testing
RICHOUX-STRICT	68 788	68 268	520	5341 duplicates, dropped 29 996 negatives for training and 200 for testing

PPIs. The positive dataset was sorted into blocks $INTRA_0$ (129 983 PPIs), $INTRA_1$ (73 714 PPIs), and $INTRA_2$ (55 322 PPIs), where $uv \in INTRA_i$ if and only if $u \in P_i$ and $v \in P_i$. Negative PPIs were sampled randomly to match the number of positives. To exclude the possibility to learn from node degrees alone, the node degrees of the proteins from the positive networks $INTRA_i$ were approximately preserved in the negative networks. This was achieved by randomly sampling two distinct proteins at a time from the multiset

$$M_i = \{\{ \underbrace{p_1, \dots p_1}_{deg_i(p_1) \text{ times}}, \dots, \underbrace{p_k, \dots p_k}_{deg_i(p_k) \text{ times}}, \dots \mid p_k \in P_i \}\} \tag{2}$$

where the number of occurences of each protein $p_k \in P_i$ equals its degree $deg_i(p_k)$ in $INTRA_i$.

Afterwards, the sequences of the individual proteins in the blocks were fed to CD-HIT at a similarity threshold of 40%. Within $INTRA_0$, CD-HIT identified 1694 redundant sequences, within $INTRA_1$ 1595, and within $INTRA_2$ 1361. Between $INTRA_0$ and $INTRA_1$, CD-HIT 2D found 17 redundant sequences, between $INTRA_0$ and $INTRA_2$ 55, and between $INTRA_1$ and $INTRA_2$ 29. These sequences were filtered out of the blocks to form redundancy-reduced datasets. The blocks were then balanced again, resulting in 140 272 PPIs in $INTRA_0$, 79 126 PPIs in $INTRA_1$, and 49 400 PPIs in $INTRA_2$. Finally, we labeled the block $INTRA_0$ as the training dataset, the block $INTRA_1$ as the validation dataset, and the block $INTRA_2$ as the test dataset.

Tested methods

The results of an extensive literature screening for high-performing PPI prediction methods can be found in Supplemental Table S1. Out of 26 reviewed publications, only nine made their code available (10 methods; Richoux et al.[2] proposed two). Five of these methods were written in Python (PIPR[4], DeepFE[13], Richoux-FC, Richoux-LSTM[2], and TransformerGO[41]), two in Matlab (You et al.[8], Ding et al.[37]), one in Lua (Hashemifar et al.[18]), and two in C++ (Hamp and Rost[49], SPRINT[31]). We excluded the methods that did not only use sequences as input (Hashemifar et al.[18], Hamp and Rost[49], TransformerGO[41]) and focused on DL methods with high reported accuracies which we managed to reproduce with reasonable effort. Additionally, we included the SPRINT method as a baseline comparison since it only relies on sequence similarity for its predictions. As further baselines, we also included simple baseline ML methods (random forest, SVM), which we ran on three different encodings for the amino acid sequences (PCA, MDS, node2vec). Overall, we hence tested five published methods and six baseline ML models.

Table 4: Number of samples contained in each block after splitting the benchmark datasets according to the partitioning assignments. n_0, n_1, and n_{INTER} denote the numbers of positive and negative PPIs in the blocks $INTRA_0$, $INTRA_1$, and *INTER*. All blocks are balanced (50% interactions, 50% non-interactions).

Dataset	n_0	n_1	n_{INTER}	Modifications
GUO	5858	1364	3744	$INTRA_0$: 859 additional negatives; $INTRA_1$: 133 deleted negatives; *INTER*: 731 deleted negatives
DU	15 918	4722	13 872	$INTRA_0$: 9859 deleted negatives; $INTRA_1$: 4114 deleted negatives; *INTER*: 14 855 deleted negatives
HUANG	2220	1496	2970	$INTRA_0$: 325 additional negatives; $INTRA_1$: 484 deleted negatives; *INTER*: 564 deleted negatives
PAN	29 318	9082	24 558	$INTRA_0$: 4955 additional negatives; $INTRA_1$: 2137 deleted negatives; *INTER*: 4853 deleted negatives
RICHOUX-UNIPROT	30 106	13 728	35 000	$INTRA_0$: 5695 additional negatives; $INTRA_1$: 4535 deleted negatives; *INTER*: 2958 deleted negatives

PIPR. PIPR[4] encodes the two input sequences using pre-trained embeddings. These embeddings take co-occurrence similarities of amino acids and physicochemical properties into account. PIPR feeds the embeddings to a deep siamese network: the embeddings are input to two residual recurrent CNN units (convolutional layer, max pooling, bidirectional GRU with residual shortcuts, global average pooling) with shared weights. The outputs are then combined using element-wise multiplication. Finally, the network predicts binary PPIs by training a multi-layer perceptron with leaky ReLU (final activation function: softmax, loss function: binary cross-entropy, optimizer: adam with AMSgrad, learning rate= 0.001, epsilon = $1e^{-6}$). For our tests, we fixed the hyperparameters according to the results of the original publication (number of epochs: 50, batch size: 256, dimension of the convolutional and GRU layers: 25, sequence embedding cropped at position: 2000).

DeepFE. DeepFE[13] pre-trained a Word2vec/Skip-gram model (size: 20, window: 4), using all Swissprot protein sequences (from 2018) as input. The model treats every amino acid as a word and, therefore, a sequence as a sentence. PPI candidate sequences are then represented using the Word2vec embeddings and are fed to a deep siamese network consisting of four units (dense layer with ReLU, batch normalization, dropout layer) with decreasing dimensionality (2048, 1024, 512, and 128). The outputs are then concatenated and put through another unit with dimensionality 8. PPI prediction is done at the final two-dimensional layer with a soft-max activation function. A stochastic gradient descent optimizer is used (learning rate: 0.01, momentum: 0.9, decay: 0.001). For our tests, we fixed the hyperparameters according to the results of the original publication (number of epochs: 45, batch size: 256, maximum protein length fed to the model: 850).

Richoux-FC and Richoux-LSTM. Richoux et al.[2] represent all sequences using one-hot encoding and present two DL models: a fully connected model and a recurrent LSTM model, which we refer to as Richoux-FC and Richoux-LSTM. Richoux-FC first flattens both sequence inputs and passes them separately through two units consisting of a dense layer (dimensionality: 20, ReLU) and a batch normalization layer.

The two outputs are concatenated and passed to another unit (dense layer, dimensionality: 20, ReLU). The final layer (dimensionality: 1) predicts the PPI using a sigmoid activation function. Richoux-LSTM first extracts features from the sequence via three units (convolution, pooling, batch normalization) and a final LSTM layer. The parameters are shared for these layers for the two input sequences. Then, the outputs are concatenated and passed to a dense layer (ReLU) with another batch normalization. Finally, the LSTM model predicts the PPIs using a sigmoid activation function on the final one-dimensional dense layer. Both models use the Adam optimizer with an initial learning rate of 0.001, which reduces by 10% if the loss plateaus for 5 epochs until a minimum of 0.0008. For our tests, we fixed the hyperparameters according to the results of the original publication (number of epochs for Richoux-FC: 25, number of epochs for Richoux-LSTM: 100, batch size: 256, maximum sequence length: 1166).

SPRINT. Instead of finding hidden patterns via DL, SPRINT[31] entirely relies on the hypothesis that a protein pair that is pairwise similar to an interacting protein pair is more likely to interact. Therefore, we can use it as a baseline model to see how well a dataset can be predicted if we only use sequence similarities (Explanation 3). SPRINT searches for similar subsequences between the candidate protein pair and the database of known interactions by using an approach very similar to BLAST's hit-and-extend approach. Instead of having consecutive initial seeds of a certain length like BLAST, SPRINT also allows for gapped initial seeds to increase the number of hits. Using all hits, SPRINT calculates similarity scores and sorts the output decreasingly by the scores where a higher score represents a higher probability for interaction. Rather than choosing an arbitrary threshold to calculate accuracies, we calculated the AUC and Area Under the Precision Recall curve (AUPR) for SPRINT.

Baseline ML models

We implemented additional baseline models to compare the performance of DL methods against classical ML. Note that the idea behind implementing the baseline models is not to show that classical ML models suffice for the PPI prediction task but to explain the phenomenal accuracies reported for sequence-based DL models. Because of this, we designed our baseline models such that they can only learn from sequence similarities and node degrees: the similarities are the only input features and the node degrees can be learnt implicitly during training.

As models, we trained a random forest (sklearn 1.0.2 RandomForestClassifier, 100 decision trees, six parallel jobs, a prediction greater than 0.5 is interpreted as interaction) and an SVM (sklearn SVC, RBF kernel, a prediction > 0.5 is interpreted as interaction). We adopted these hyperparameters from the sklearn defaults; no tuning was done.

Proteins were represented using a similarity vector where each entry is the pairwise similarity score (bitscore) to every other protein in the human or yeast proteome (defined by Swissprot). A PPI was encoded as the concatenation of the two embeddings. An all-against-all similarity matrix for human and yeast was computed by the team of SIMAP2[34], yielding a 6718×6718 matrix for yeast and a $20\,353 \times 20\,353$ matrix for human. Since this dimensionality is too high, we reduced it via three approaches: PCA, MDS, and node2vec, each returning 128 dimensions.

Theoretically, preprocessing the whole dataset before splitting it into train and test is considered data leakage. In this case, it is acceptable since we consider the whole universe (proteome). Introducing data leakage via joint preprocessing usually means that we have information available during testing that we would not have in a real scenario (e.g., divergence from the mean of the dataset). However, the dimensionality reductions are computed on the similarity matrix of all known human proteins; therefore, we do not introduce any dataset-specific biases. Given a new PPI, we would already possess the embeddings of its individual proteins.

PCA and MDS yielded matrices of dimensions 6718×128 and $20\,353 \times 128$ for the yeast and human proteomes. For the computation of the node2vec encoding, we built a weighted similarity network from

the similarity matrix such that an edge between two proteins exists if and only if their bitscore is positive. Self-loops were excluded. Since SIMAP2 uses cutoffs, some of the proteins had no edges. Because the input to node2vec is just an edge list, the resulting node2vec embeddings were smaller, resulting in matrices of dimensions 6194×128 and $20\,210 \times 128$ for the yeast and human proteomes. Altogether, we hence tested three different embeddings as input to two ML models, yielding six baseline models.

Code and data availability

All datasets and code are available at `https://github.com/biomedbigdata/data-leakage-ppi-prediction`. The gold standard dataset is available at `https://doi.org/10.6084/m9.figshare.21591618.v1`. An AIMe report[50] specifying the details of all analyses is available at `https://aime-registry.org/report/VRPXym`.

Acknowledgments

We thank Prof. Dr. Thomas Rattei and his team from SIMAP2 at the University of Vienna for kindly and quickly sharing their protein similarity data.

References

[1] Rao, V. S., Srinivas, K., Sujini, G. & Kumar, G. Protein-protein interaction detection: methods and analysis. *International journal of proteomics* **2014** (2014).

[2] Richoux, F., Servantie, C., Borès, C. & Téletchéa, S. Comparing two deep learning sequence-based models for protein-protein interaction prediction. *arXiv preprint arXiv:1901.06268* (2019).

[3] Sun, T., Zhou, B., Lai, L. & Pei, J. Sequence-based prediction of protein protein interaction using a deep-learning algorithm. *BMC bioinformatics* **18**, 1–8 (2017).

[4] Chen, M. *et al.* Multifaceted protein–protein interaction prediction based on siamese residual rcnn. *Bioinformatics* **35**, i305–i314 (2019).

[5] Wang, L., Wang, H.-F., Liu, S.-R., Yan, X. & Song, K.-J. Predicting protein-protein interactions from matrix-based protein sequence using convolution neural network and feature-selective rotation forest. *Scientific reports* **9**, 1–12 (2019).

[6] Xu, D., Xu, H., Zhang, Y., Chen, W. & Gao, R. Protein-protein interactions prediction based on graph energy and protein sequence information. *Molecules* **25**, 1841 (2020).

[7] Wang, J., Zhang, L., Jia, L., Ren, Y. & Yu, G. Protein-protein interactions prediction using a novel local conjoint triad descriptor of amino acid sequences. *International Journal of Molecular Sciences* **18**, 2373 (2017).

[8] You, Z.-H., Chan, K. C. & Hu, P. Predicting protein-protein interactions from primary protein sequences using a novel multi-scale local feature representation scheme and the random forest. *PloS one* **10**, e0125811 (2015).

[9] You, Z.-H. *et al.* Detecting protein-protein interactions with a novel matrix-based protein sequence representation and support vector machines. *BioMed research international* **2015** (2015).

[10] You, Z.-H., Lei, Y.-K., Zhu, L., Xia, J. & Wang, B. Prediction of protein-protein interactions from amino acid sequences with ensemble extreme learning machines and principal component analysis. *BMC bioinformatics* **14**, 1–11 (2013).

[11] Hu, L. & Chan, K. C. Discovering variable-length patterns in protein sequences for protein-protein interaction prediction. *IEEE transactions on nanobioscience* **14**, 409–416 (2015).

[12] Du, X. *et al.* Deepppi: boosting prediction of protein–protein interactions with deep neural networks. *Journal of chemical information and modeling* **57**, 1499–1510 (2017).

[13] Yao, Y., Du, X., Diao, Y. & Zhu, H. An integration of deep learning with feature embedding for protein–protein interaction prediction. *PeerJ* **7**, e7126 (2019).

[14] Jha, K. & Saha, S. Amalgamation of 3d structure and sequence information for protein–protein interaction prediction. *Scientific Reports* **10**, 1–14 (2020).

[15] Saha, I. *et al.* Ensemble learning prediction of protein–protein interactions using proteins functional annotations. *Molecular BioSystems* **10**, 820–830 (2014).

[16] Chen, K.-H., Wang, T.-F. & Hu, Y.-J. Protein-protein interaction prediction using a hybrid feature representation and a stacked generalization scheme. *BMC bioinformatics* **20**, 1–17 (2019).

[17] Zhao, L., Wang, J., Hu, Y. & Cheng, L. Conjoint feature representation of go and protein sequence for ppi prediction based on an inception rnn attention network. *Molecular Therapy-Nucleic Acids* **22**, 198–208 (2020).

[18] Hashemifar, S., Neyshabur, B., Khan, A. A. & Xu, J. Predicting protein–protein interactions through sequence-based deep learning. *Bioinformatics* **34**, i802–i810 (2018).

[19] Maetschke, S. R., Simonsen, M., Davis, M. J. & Ragan, M. A. Gene ontology-driven inference of protein–protein interactions using inducers. *Bioinformatics* **28**, 69–75 (2012).

[20] Khatun, M., Shoombuatong, W., Hasan, M., Kurata, H. *et al.* Evolution of sequence-based bioinformatics tools for protein-protein interaction prediction. *Current Genomics* **21**, 454–463 (2020).

[21] Jumper, J. *et al.* Highly accurate protein structure prediction with alphafold. *Nature* **596**, 583–589 (2021).

[22] Evans, R. *et al.* Protein complex prediction with alphafold-multimer. *BioRxiv* (2021).

[23] Park, Y. & Marcotte, E. M. Flaws in evaluation schemes for pair-input computational predictions. *Nature methods* **9**, 1134–1136 (2012).

[24] Hamp, T. & Rost, B. More challenges for machine-learning protein interactions. *Bioinformatics* **31**, 1521–1525 (2015).

[25] Li, S. *et al.* Recent advances in predicting protein–protein interactions with the aid of artificial intelligence algorithms. *Current Opinion in Structural Biology* **73**, 102344 (2022).

[26] Whalen, S., Schreiber, J., Noble, W. S. & Pollard, K. S. Navigating the pitfalls of applying machine learning in genomics. *Nature Reviews Genetics* **23**, 169–181 (2022).

[27] Kapoor, S. & Narayanan, A. Leakage and the reproducibility crisis in ML-based science. *arXiv preprint arXiv:2207.07048* (2022).

[28] Kaufman, S., Rosset, S., Perlich, C. & Stitelman, O. Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data (TKDD)* **6**, 1–21 (2012).

[29] Ben-Hur, A. & Noble, W. S. Choosing negative examples for the prediction of protein-protein interactions. *BMC Bioinformatics* **7**, 1–6 (2006).

[30] Berggård, T., Linse, S. & James, P. Methods for the detection and analysis of protein–protein interactions. *Proteomics* **7**, 2833–2842 (2007).

[31] Li, Y. & Ilie, L. Sprint: ultrafast protein-protein interaction prediction of the entire human interactome. *BMC bioinformatics* **18**, 1–11 (2017).

[32] Guo, Y., Yu, L., Wen, Z. & Li, M. Using support vector machine combined with auto covariance to predict protein–protein interactions from protein sequences. *Nucleic acids research* **36**, 3025–3030 (2008).

[33] Pan, X.-Y., Zhang, Y.-N. & Shen, H.-B. Large-scale prediction of human protein- protein interactions from amino acid sequence based on latent topic features. *Journal of proteome research* **9**, 4992–5001 (2010).

[34] Arnold, R., Goldenberg, F., Mewes, H.-W. & Rattei, T. Simap—the database of all-against-all protein sequence similarities and annotations with new interfaces and increased coverage. *Nucleic acids research* **42**, D279–D284 (2014).

[35] Huang, Y.-A., You, Z.-H., Gao, X., Wong, L. & Wang, L. Using weighted sparse representation model combined with discrete cosine transformation to predict protein-protein interactions from protein sequence. *BioMed research international* **2015** (2015).

[36] Wang, L. *et al.* Using two-dimensional principal component analysis and rotation forest for prediction of protein-protein interactions. *Scientific reports* **8**, 1–10 (2018).

[37] Ding, Y., Tang, J. & Guo, F. Predicting protein-protein interactions via multivariate mutual information of protein sequences. *BMC bioinformatics* **17**, 1–13 (2016).

[38] Guo, Y. *et al.* Pred_ppi: a server for predicting protein-protein interactions based on sequence data with probability assignment. *BMC research notes* **3**, 1–7 (2010).

[39] Shen, J. *et al.* Predicting protein–protein interactions based only on sequences information. *Proceedings of the National Academy of Sciences* **104**, 4337–4341 (2007).

[40] Mahapatra, S., Kumar, A., Sharma, A. & Sahu, S. S. Effect of dimensionality reduction on classification accuracy for protein–protein interaction prediction. In *Advanced Computing and Intelligent Engineering*, 3–12 (Springer, 2020).

[41] Ieremie, I., Ewing, R. M. & Niranjan, M. Transformergo: predicting protein–protein interactions by modelling the attention between sets of gene ontology terms. *Bioinformatics* **38**, 2269–2277 (2022).

[42] Sanders, P. & Schulz, C. Think locally, act globally: Highly balanced graph partitioning. In *International Symposium on Experimental Algorithms*, 164–175 (Springer, 2013).

[43] Soleymani, F., Paquet, E., Viktor, H., Michalowski, W. & Spinello, D. Protein–protein interaction prediction with deep learning: A comprehensive review. *Computational and Structural Biotechnology Journal* (2022).

[44] Bernett, J. PPI prediction from sequence, gold standard dataset. figshare (2022). `https://figshare.com/articles/dataset/PPI_prediction_from_sequence_gold_standard_dataset/21591618`. `https://doi.org/10.6084/m9.figshare.21591618.v1`.

[45] Alanis-Lobato, G., Andrade-Navarro, M. A. & Schaefer, M. H. Hippie v2. 0: enhancing meaningfulness and reliability of protein–protein interaction networks. *Nucleic acids research* gkw985 (2016).

[46] Fu, L., Niu, B., Zhu, Z., Wu, S. & Li, W. CD-HIT: accelerated for clustering the next-generation sequencing data. *Bioinformatics* **28**, 3150–3152 (2012). `https://doi.org/10.1093/bioinformatics/bts565`. `https://doi.org/10.1093/bioinformatics/bts565`. `https://academic.oup.com/bioinformatics/article-pdf/28/23/3150/18529929/bts565.pdf`.

[47] Louadi, Z. *et al.* Digger: exploring the functional role of alternative splicing in protein interactions. *Nucleic acids research* **49**, D309–D318 (2021).

[48] Blohm, P. *et al.* Negatome 2.0: a database of non-interacting proteins derived by literature mining, manual annotation and protein structure analysis. *Nucleic acids research* **42**, D396–D400 (2014).

[49] Hamp, T. & Rost, B. Evolutionary profiles improve protein–protein interaction prediction from sequence. *Bioinformatics* **31**, 1945–1950 (2015).

[50] Matschinske, J. *et al.* The AIMe registry for artificial intelligence in biomedical research. *Nat. Methods* **18**, 1128–1131 (2021). `https://doi.org/10.1038/s41592-021-01241-0`.