# Using Deep Learning to Annotate the Protein Universe

Maxwell L. Bileschi[1, *], David Belanger[1], Drew Bryant[1], Theo Sanderson[1], Brandon Carter[1,2], D. Sculley[1], Mark A. DePristo[1], and Lucy J. Colwell[1, 3, *]

[1]Google Research
[2]Computer Science and Artificial Intelligence Laboratory, MIT
[3]Dept. of Chemistry, Cambridge University
[*]Correspondence to mlbileschi@google.com and lcolwell@google.com

May 6, 2019

**Abstract**

Understanding the relationship between amino acid sequence and protein function is a long-standing problem in molecular biology with far-reaching scientific implications. Despite six decades of progress, state-of-the-art techniques cannot annotate 1/3 of microbial protein sequences, hampering our ability to exploit sequences collected from diverse organisms. To address this, we report a deep learning model that learns the relationship between unaligned amino acid sequences and their functional classification across all 17929 families of the Pfam database. Using the Pfam seed sequences we establish a rigorous benchmark assessment and find a dilated convolutional model that reduces the error of both BLASTp and pHMMs by a factor of nine. Using 80% of the full Pfam database we train a protein family predictor that is more accurate and over 200 times faster than BLASTp, while learning sequence features it was not trained on such as structural disorder and transmembrane helices. Our model co-locates sequences from unseen families in embedding space, allowing sequences from novel families to be accurately annotated. These results suggest deep learning models will be a core component of future protein function prediction tools.

Predicting the function of a protein from its raw amino acid sequence is the critical step for understanding the relationship between genotype and phenotype. As the cost of DNA sequencing drops and metagenomic sequencing projects flourish, fast and efficient tools that annotate open reading frames with function will play a central role in exploiting this data [1, 2]. Identifying proteins that catalyze novel reactions, bind specific microbial targets or work together to build new molecules will accelerate advances in biotechnology. Current practice for functional prediction of a novel protein sequence involves alignment across a large database of annotated sequences using algorithms such as BLASTp [3], or

profile hidden Markov models (pHMMs) built from aligned sequence families such as those provided by Pfam [4, 5].

While these approaches are generally successful, at least one-third of microbial proteins cannot be annotated through alignment to characterized sequences [6, 7]. Moreover, the run times of methods such as BLASTp scale nearly linearly with the size of the labelled database, which is growing exponentially [8]. Broad protein families require muliple HMM profiles to model their diversity [9], while more than 22% of the highly-curated families in Pfam 32.0 have no functional annotation. More generally, models that predict function from sequence are limited by pipelines that require substitution matrices, sequence alignment, and hand-tuned scoring functions.

Deep learning provides an opportunity to bypass these bottlenecks and directly predict protein functional annotations from sequence data. In these frameworks, a single model learns the distribution of multiple classes simultaneously, and can be rapidly evaluated. Besides providing highly accurate models, the intermediate layers of a deep neural network trained with supervision can capture high-level structure of the data through learned representations [10]. These can be leveraged for exploratory data analysis or supervised learning on new tasks, in particular those with limited data. For example, novel classes can be identified from just a few examples through few-shot learning.

This raises the question of whether deep learning can provide protein function prediction tools with broad coverage of the protein universe, as found in the 17929 families of the recent Pfam 32.0 release [11]. Recent work that applies deep learning is either restricted to regimes that are not practical or representative in terms of the number of families or required family size, or does not provide comparison to existing approaches that enjoy widespread use. For example, DeepSF classifies sequences into 1195 SCOP fold classes [12, 13], while DeepFam [14] considers 2892 COG families of more than 100 sequences each. SECLAF uses hundreds of SwissProt classes with more than 150 sequences to train a deep model [15–17]. DEEPre uses sequence and Pfam annotations to predict enzyme commission (EC) classes [18, 19]. D-SPACE reports powerful embeddings learned by a deep model, but does not compare classification performance with existing methods [20]. In [21] a graph convolutional network reduces the required family size to 20 sequences, reporting 58% accuracy. It is encouraging that novel deep learning predictions for four functional classes were experimentally validated in [22].

Building confidence in deep learning approaches requires benchmarks that enable fair and rigorous comparison with existing state of the art methods and among deep model architectures. In this paper we pose protein function prediction as a sequence annotation task. We use 1.34 million Pfam `seed` sequences to construct a benchmark, and assess the performance of deep models at annotating unaligned protein domain sequences. This benchmark has an order of magnitude both more families and fewer examples per family than previous deep learning efforts (Supplementary Fig. 1). We show that the deep models are substantially faster at annotating held-out test sequences than state of the art profile HMM and BLASTp approaches, and reduce the error rate almost ten-fold. We use the joint

representation learned across protein families in one-shot learning to annotate sequences from small families that the model was not trained on. These findings support claims that deep learning models have the potential to provide a general solution to the challenge of protein functional annotation, and accelerate our ability to understand and exploit metagenomic sequence data.

## Results

We use the Pfam `seed` dataset to construct a benchmark[1] and compare the performance of deep learning models with existing methods including BLASTp, phmmer and HMMER. ProtENN uses a simple majority vote across an ensemble of 13 Deep CNN (ProtCNN) models to achieve a predictive accuracy of 99.84%, reducing both the HMM and BLASTp error rates by a factor of 9, to 201 misclassified sequences (Table 1, Supplementary Fig. 2). Fig. 1 shows that both ProtCNN and ProtENN generalize well to sequences that are distinct from those in the training set. A single ProtCNN model has an error rate of 0.5%, reducing the HMM and BLASTp error rates threefold. Furthermore the HMMs as implemented by HMMER 3.1b yield no prediction for 445 sequences (0.35%) of the test set, increasing the number of errors to 2229. Enforcing the Pfam curated gathering thresholds for family membership would return multiple above-threshold hits for 8.5% of test sequences, further decreasing the reported HMM accuracy.

Overall, the HMM makes multiple errors for both large and small Pfam families, while ProtENN makes single errors for 151 of the 164 families where it falters. Specifically, the 201 sequences misclassified by ProtENN are drawn from 164 families of average size 141, while a single ProtCNN misclassifies 625 sequences from 550 families. In contrast, the 1784 errors made by the HMM are drawn from 392 families of average size 1091. Fig. 2 shows error rates for families where both the HMM and ProtENN make mistakes. We find 11 sequences that are consistently predicted incorrectly in exactly the same way by all ensemble elements of ProtENN. Supplementary Table 6 suggests that there is some ambiguity about their correct family label in each case. For example, our models predict that the sequence R7EGP4_9BACE/13-173 from Pfam family DUF1282, actually belongs to the YIP1 family. The hhsearch [4] tool predicts that DUF1282 is similar to the Yip1 family, while, BLASTp finds that this sequence is identical to sequences annotated as the YIP1 family, agreeing with the ProtENN prediction.

### Sequence Annotation for Pfam `full`

The predictive accuracy of deep learning models typically improves as the amount of well-labelled training data increases. Likewise, accuracy of nearest-neighbor methods models

[1] Available for download at `https://console.cloud.google.com/storage/browser/brain-genomics-public/research/proteins/pfam/random_split`, interactive notebook at `https://www.kaggle.com/googleai/pfam-seed-random-split`

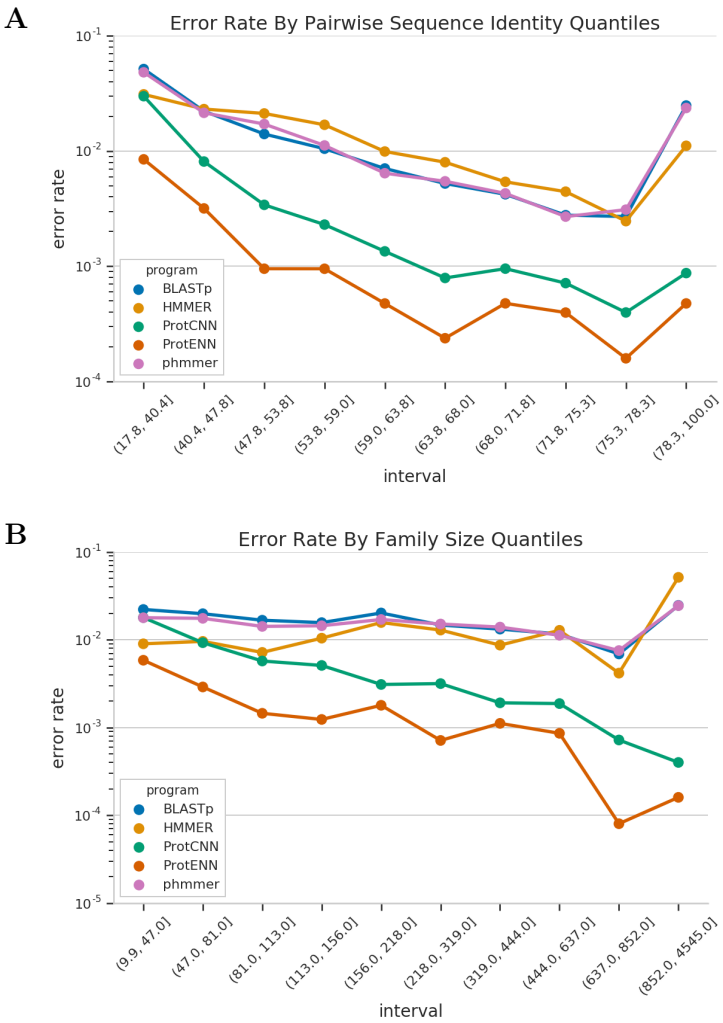


Figure 1: Comparison of model error rate (log scale) on the held-out test set for Pfam `seed` sequence data as a function of (A) sequence identity (for the highest scoring pair found in the training set as reported by BLASTp), and (B) as a function of family size. ProtENN performs substantially better than all other models across all distances and family sizes. Additional breakdowns of this data are available in Supplementary Fig. 3.

| Model | Error rate | Number of errors |
|---|---|---|
| HMM Top Pick | 1.414% | 1784 |
| phmmer | 1.531% | 1932 |
| BLASTp | 1.654% | 2087 |
| k-mer | 9.994% | 12610 |
| RNN | 1.800% | 2271 |
| 1-ResNet block CNN | 1.120% | 1413 |
| 2-ResNet block CNN | 0.852% | 1075 |
| ProtCNN | 0.495% | 625 |
| **ProtENN** | **0.159%** | **201** |

Table 1: Performance on randomly-split data. For additional breakdown of this data see Supplementary Figs. 3 and 4.

such as BLASTp also improves, but at the expense of computational performance. To compare these approaches on a larger dataset, we use the 54.5 million sequences of the Pfam `full` database [23] and follow the protocol established for the `seed` benchmark: we randomly split each family, assigning 80% of sequences to the train set and 10% each to dev and test sets, and carry out a hyperparameter search to optimize ProtCNN accuracy for this new task. To provide a highly accurate baseline we impute labels via the top BLASTp hit, using the training set as the query database.

Our resulting ProtCNN model has an error rate of just 1.26% (∼69k errors), lower than the BLASTp error rate of 1.78% (∼97k errors). ProtENN, ensembled across 13 ProtCNN models, reduces the error even further to just 0.5% (∼25k errors). Fig. 3 shows that ProtENN is more accurate than BLASTp across all deciles of sequence identity, and also for sequences from all but the smallest 10% of families. This reduction in error rate is particularly attractive when coupled with the gains in computational performance achieved by ProtENN.

## Computational Performance

Once trained, the deep learning models provide substantial speed improvements for protein domain sequence annotation. Table 2 compares their computational performance with existing methods on our Pfam `seed` benchmark. ProtCNN is almost twice as fast as HMMER 3.1b, the fastest (though not the most accurate) existing method.

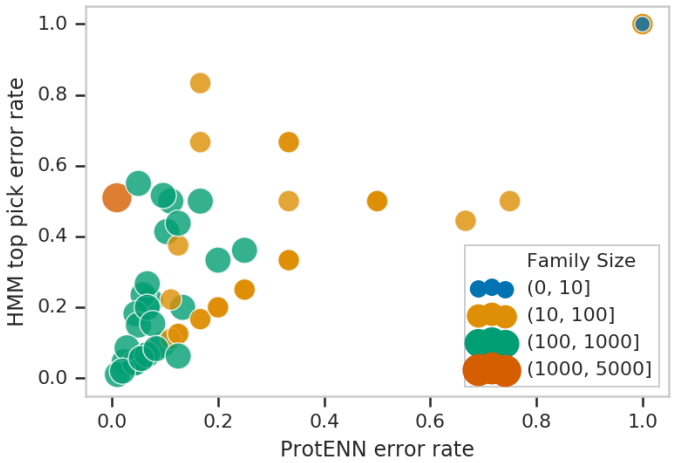


Figure 2: Comparison of the 67 families for which there was at least one error for both ProtENN and HMM top pick for the Pfam `seed` dataset. ProtENN achieves better performance for families that lie above the line y=x, while the opposite is true for those families that fall below this line. HMM top pick tends to have higher error rates for larger families. Additional details are available in Supplementary Fig. 4 and Supplementary Table 8.

| | Inferences per second | Time to predict all 1.34 million Pfam `seed` sequences |
|---|---|---|
| phmmer | 2.52 | 6.15 days |
| HMMER | 150.20 | 2.48 hours |
| HMMER (no filters) | 0.66 | 23.58 days |
| BLASTp | 11.27 | 15.2 hours |
| **ProtCNN** | **247.39** | **1.5 hours** |

Table 2: Inference speed comparison of different models trained on the Pfam `seed` training set. BLASTp was run on a 96 core CPU, while ProtCNN was run on a P100 GPU (See Supplementary Table 7 for hardware details).

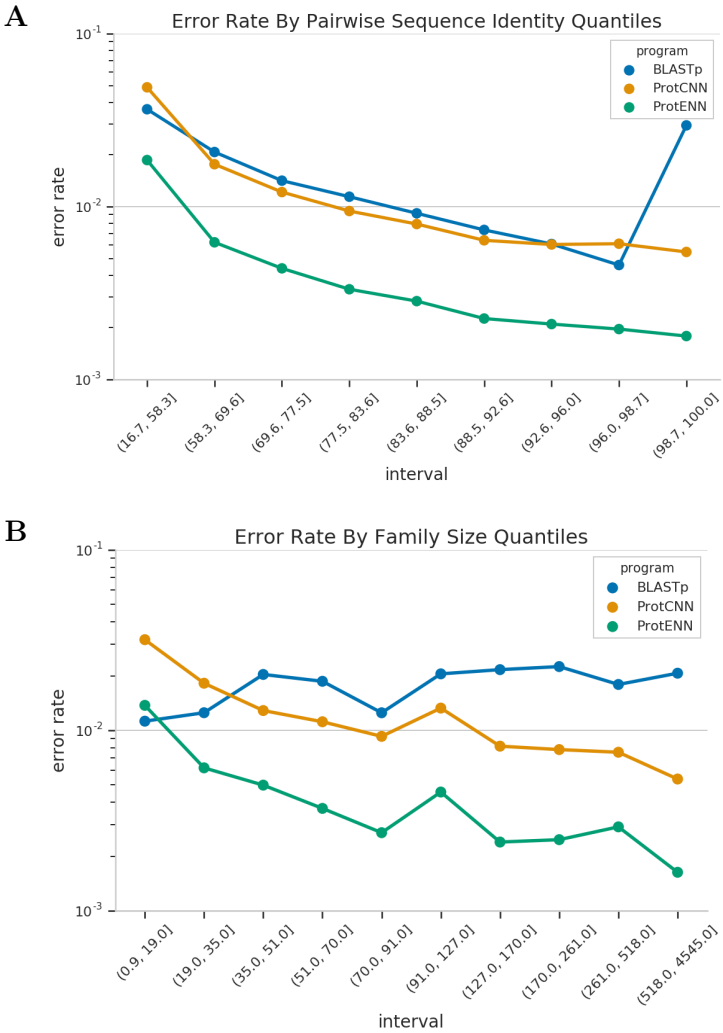


Figure 3: Comparison of model error rate on the held out test set for Pfam `full` sequence data (A) as a function of sequence identity (for highest scoring pair found in the training set as reported by BLASTp), and (B) as a function of family size.

The BLASTp calculation for the Pfam `full` test set (∼10% of Pfam `full`) takes 34 days on a 96 core CPU, while computing all test predictions for a single ProtCNN model takes less than 3.6 hours on a single GPU. Our most accurate ProtCNN model for Pfam `full` has just a single ResNet block, and as a result it is more than 200 times faster and more accurate than BLASTp. The estimated prediction times for the 54.5 million sequences of the Pfam `full` dataset are given in Table 3.

| | Inferences per second | Time to predict all 54.5 million Pfam `full` sequences |
|---|---|---|
| BLASTp | 1.85 | 340 days |
| **ProtCNN** | **415.74** | **36 hours** |

Table 3: Inference speed comparison of different models trained on the Pfam `full` training set. BLASTp was run on a 96 core CPU, while ProtCNN was run on a NVIDIA P100 GPU.

## What does ProtCNN learn?

To interrogate what ProtCNN learns about the natural amino acids, we add a 5-dimensional trainable representation between the one-hot amino acid input and the embedding network (see Methods for details), and retrain our ProtCNN model on the same unaligned sequence data from Pfam `full`, achieving the same performance. Fig. 4A (left) shows the cosine similarity matrix of the resulting learned embedding, while Fig. 4A (right) shows the BLOSUM62 matrix, created using aligned sequence blocks at roughly 62% identity [24]. The structural similarities between these matrices suggest that ProtCNN has learned known amino acid substitution patterns from the unaligned sequence data.

We next ask whether ProtCNN can distinguish between variants of the same protein domain sequence with single amino acid substitutions, despite the lack of residue-level supervision during training. To measure the predicted impact of sequence changes, we use a single ProtCNN trained on Pfam `full` to calculate the model's predicted distribution over classes for the original and modified sequences. We then compute the KL-divergence between these two probability distributions to quantify the effect of the subsitution on the model prediction. Fig. 4B reports this measure for every possible single amino acid substitution within an ATPase domain sequence. Most substitutions in the disordered region are predicted to have negligible effect, with the exception of mutations to phenylalanine, tyrosine and tryptophan, amino acids that are known to promote order [25]. This ATPase domain also contains two transmembrane helices, within which the order of amino acid (using IUPAC amino acid codes) preference according to ProtCNN is `FMLVI YACTS WGQHN KRPED`. The suggestion that charged amino acids and proline are avoided within these regions again agrees with existing knowledge. An additional example saturation mutagenesis prediction is shown in Supplementary Fig. 5.

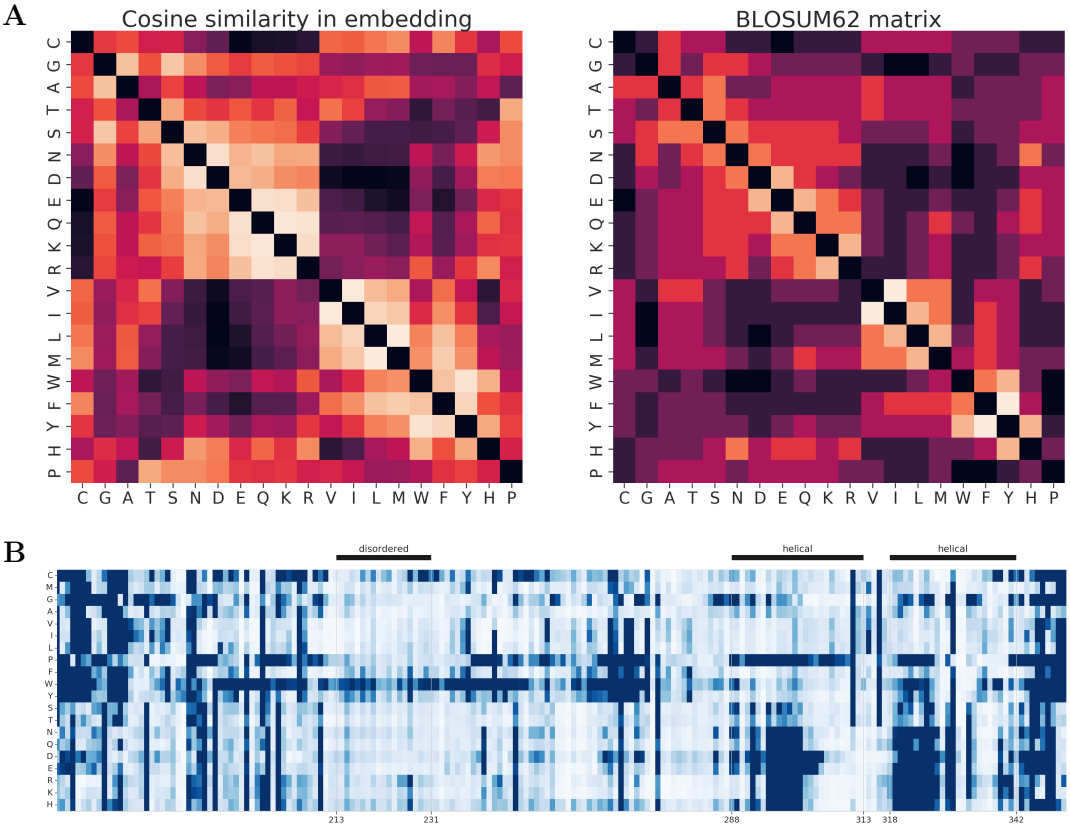


Figure 4: (A) The amino acid embedding extracted from the trained ProtCNN model yields cosine similarities in embedding space that reflect the overall structure of the BLOSUM62 matrix [24]. (B) Predicted change in function for each missense mutation in ATPase domain AT1A1_PIG/161-352 from family PF00122.20. The ProtCNN model (trained using Pfam `full`) appropriately predicts that most substitutions in the disordered region are unlikely to change the protein's function. Substitutions to phenylalanine, tyrosine and tryptophan are predicted to have the largest effect on function within the disordered region, in agreement with their known order-promoting properties [25]. The wild type sequence is available in Supplemental Table 9.

## One-Shot Sequence Classification

Finally, we show that ProtCNN can accurately classify sequences from families that the model has not been trained on. This is motivated by the biologically important question of novel family identification, where each novel family is anchored by a single founder sequence. Taking all but the final layer of ProtCNN generates a map from the space of protein sequence to a 1100-dimensional embedding space. Since the model is trained for classification, sequences from different families should be well-separated by this map. This provides a variety of opportunities, such as annotation of domains of unknown function and supervised learning on small datasets [20, 26]. Here, we proceed by training a ProtCNN model on the 12361 Pfam `seed` families with more than 9 training sequences. The remaining 5568 families consist of 710 families that have a single held out test sequence, and the 4858 smallest families that have no test sequences (because they were so small, see Methods). By construction, this ProtCNN model achieves 0% accuracy on the 710 test sequences from families not seen during training.

| Prediction Method | Families Included in Training | Overall Error Rate | Small Family Error Rate | Large Family Error Rate |
|---|---|---|---|---|
| ProtCNN | Large only | 1.0% | 100.0% | 0.4% |
| One-Shot 1-NN | | 0.8% | 14.9% | 0.7% |
| Two-Shot 1-NN | | 0.8% | 9.0% | 0.7% |
| Per-Family 1-NN | | 0.7% | 1.3% | 0.7% |
| Per-Instance 1-NN | | 0.6% | 2.4% | 0.6% |
| ProtCNN | All | 0.4% | 3.0% | 0.4% |
| Per-Family 1-NN | | 0.7% | 0.8% | 0.7% |
| Per-Instance 1-NN | | 0.5% | 2.3% | 0.5% |

Table 4: Performance when classifying using nearest neighbors in embedding space. Resolution on error rates is limited by the number of sequences in the test set (710).

We compute an average embedding for each of the 12361 large training families, and embed a single training sequence from each of the 5568 small families held out from the model. This yields a representative embedding for each of the 17929 families in Pfam 32.0. We then use proximity in this space to carry out one-shot learning for the smallest classes, which were most difficult for ProtCNN to annotate. Specifically, for each of the 710 test sequences, we perform nearest-neighbor classification (*Per-Family 1-NN*) using cosine similarity in embedding space with the set of representatives. Table 4 shows that this approach achieves a one-shot classification error rate of 15%. Increasing to two-shot learning results in an error rate of 9%.

In Per-Family 1-NN we use all available training examples for the small classes to

construct per-family embeddings. In contrast, *Per-Instance 1-NN* finds the nearest neighbor for each test sequence among the embeddings of every training sequence. Table 4 shows that Per-Family 1-NN is particularly powerful at accurately classifying sequences from small families. In the final three rows, we provide an upper bound for performance by training ProtCNN on data from all families, and repeating the experiments described above. Again, the Per-Family 1-NN approach outperforms both ProtCNN and Per-Instance 1-NN at accurately classifying sequences from small families. We speculate that Per-Family 1-NN performs better than Per-Instance 1-NN due to noise reduction that results from averaging.

## Discussion

Both ProtCNN and ProtENN produce state of the art accuracy for protein domain annotation on a benchmark built from Pfam `seed` that is representative of known protein sequence space. With just under 1.1 million training examples across 17929 output families of vastly different sizes (Supplementary Fig. 1) our ProtENN models are highly accurate despite having no access to the alignments used to train the HMMs. These results present a significant advance over prior work; for example, 8727 families from the benchmark have at least one test sequence and fewer than the minimum 67 training examples used for classification by DeepFam [14]. Our ProtENN trained across Pfam `full` yields a three-fold reduction in the error rate and at least an order of magnitude improvement in computational efficiency compared to BLASTp.

We also show that the representation of protein sequence space learned by ProtCNN can be used in one-shot learning to classify sequences from unseen protein families. This suggests an iterative approach to novel family construction, in which a single founder sequence is used to find additional family members, which are used to update the average embedding for this putative new family and so forth, inspired by current methods such as PSI-BLAST [3], jackhmmer [27, 28] and hhblits [4]. This result suggests that a deep model trained on an existing corpus of data (here the training sequences from large Pfam seeds) can build a new family from a single sequence. Future work will test this approach beyond the confines of the benchmark Pfam `seed` dataset.

Our ProtENN models achieve extremely high accuracy without prior knowledge of protein sequence properties encoded through substitution matrices, sequence alignment or hand-curated scoring functions. Additionally, the trained models enable new sequences to be labelled much more quickly than using existing state of the art alignment-based approaches. The embedding network in each ProtCNN model maps an input sequence to a single vector representation that alone can be used for accurate family classification, pairwise sequence comparison or other downstream analysis. This differs substantially from approaches such as BLASTp, phmmer and HMMER that perform classification using explicit alignment. We note that simpler models provide useful attribution of model decision making, and we anticipate that similar insights will emerge from work that improves the interpretation and

understanding of deep learning models [29–31].

In this work, we focus on protein domain sequence annotation to provide a benchmark with broad coverage that enables comparison with the state of the art profile HMMs provided by Pfam 32.0. Using this benchmark, we report that ResNet-based ProtCNN models are faster and more accurate than current approaches, and that simple ProtCNN model ensembles provide increased accuracy. The performance of ProtCNN is most clearly limited by memory footprint, a barrier that can be overcome with additional computational resources. The model training protocol that we describe can be applied to any set of labelled protein sequence data; while the Pfam database is carefully curated, at least 25% of sequences have no experimentally validation function [11], and additional experimental functional characterization of protein sequences would be highly valuable. Our results suggest that deep learning models can rapidly and efficiently annotate novel protein sequences, such approaches have the potential to unlock novel molecular diversity for both therapeutic and biotechnology applications.

# Methods

## Deep Learning Models

We use unaligned sequence data to train deep learning models that learn the distribution across protein families through joint optimization of a softmax regression loss function. The input network maps a sequence of $L$ amino acids to an $L \times 20$ binary array, where each column is a one-hot amino acid representation. Sequences are padded to the length of the longest sequence in the batch with all-zero vectors on the right (Fig. 5A). The embedding neural network maps the $L \times 20$ array containing the one-hot amino acid representation of the sequence to an $L \times F$ array that contains an embedding for each sequence residue (see Fig. 5B). For residues outside the set of the 20 natural amino acids, we use a column of zeros. All processing in the the subsequent embedding network is designed such that it is invariant to the padding that was introduced for a given sequence. Details of neural network hyperparameters that were tuned using the development set are provided in Supplementary Tables 2, 3, 4 and 5, and used in Supplementary Figure 2. Fig. 5 depicts the input, embedding and prediction networks that make up each deep learning model. The input and prediction networks have the same functional form for all models, while the embedding network varies.

Our ProtCNN networks use residual networks (ResNets [32], a variant of convolutional neural networks that train faster and are more stable, even with many layers [32]). Fig. 5C depicts the ResNet architecture, which includes dilated convolutions[33]. The ProtCNN networks are translationally invariant, an advantage for working with unaligned protein sequence data. Convolutional architectures build up layered representations spanning many residues. An n-dilated 1d-convolution takes standard convolution operations over every nth element in a sequence, allowing local and global information to be combined without greatly increasing the number of model parameters. An important composite hyperparameter is the *receptive field size* of each per-residue feature, which describes the length of the subsequence that affects its value. Using dilated convolutions enables larger receptive field sizes to be obtained without an explosion in the number of model parameters. For this rigorous benchmark setup, Supplementary Fig. 2 suggests that larger receptive fields correspond to higher accuracy values (though see the note below about memory footprint). To our knowledge, this is the first application of dilated convolutions to protein sequence classification.

The $L \times F$ array is then pooled along the length of the sequence ensuring invariance to padding. The prediction network maps the output of the embedding network $F$ to a distribution over labels using a multi-class logistic regression model, where the vector of probabilities is obtained as $\text{SoftMax}(Wf + b)$ and $W$ and $b$ are learned weights and biases. The model prediction is the most likely label under this distribution. At train time, the log-likelihood and its gradient with respect to the parameters of the prediction and embedding networks are computed using standard forward and back propagation.

ProtCNN is orders of magnitude faster at making predictions than BLASTp; the basic

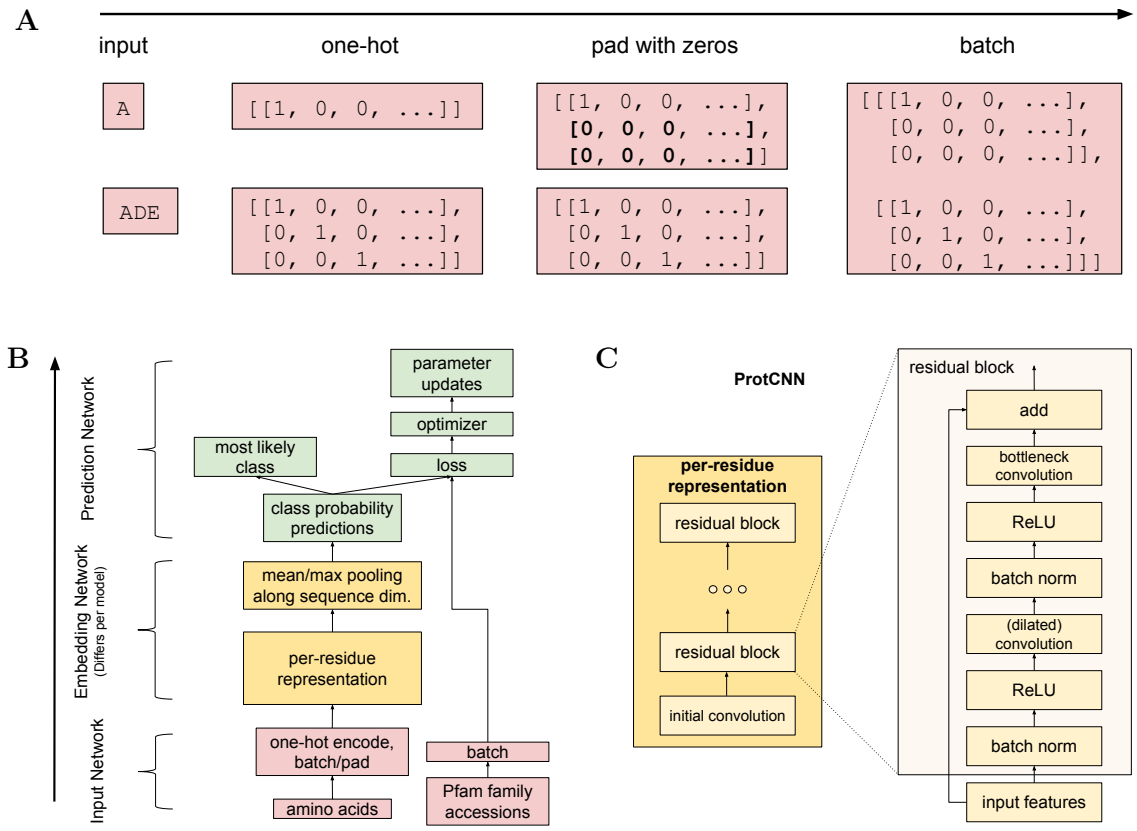


Figure 5: Architecture descriptions of neural networks. (A) Adding padding to a sequence for batching. (B) The model architecture shared by all neural networks includes the Input (red), Embedding (yellow), and Prediction (green) Networks. (C) ResNet architecture used by the ProtCNN models.

numerical operations required can be parallelized both along the length of the sequence and across multiple sequences, and can be accelerated on modern hardware. In addition to the CNN models we also trained a recurrent neural network (RNN) with single-layer bidirectional LSTM [34], which achieved accuracy of 0.982 on the Pfam `seed` dataset. Further details are provided in the Supplementary Information. Replicate Deep CNN models trained on different orderings of the same data with different random parameter initializations were found to make distinct errors, which led us to use an ensemble of ProtCNN models that we call ProtENN.

Overall those ProtCNN models that perform best tend to have the largest memory footprint, to some extent irrespective of how that memory footprint is achieved. Increasing the number of model parameters via the number of filters, the kernel size and/or the number of ResNet blocks, and increasing the batch size can all produce performance improvements. Fundamentally, the memory footprint of the models we trained was limited by the amount of memory available on a single GPU, necessitating trade offs among these different factors. We didn't explore TPUs, multiple GPUs or CPUs, all of which could result in better models. This suggests that there is headroom for future machine learning developments on this task. Among the experiments that we ran, perhaps surprisingly, the best performing ProtCNN for Pfam `full` consisted of a single residual block with 2000 filters, a kernel size of 21, and a batch size of 64.

## Model Training

We use the Adam optimizer [35]. The learning rate is subject to exponential decay following a warm-up period, although the warm-up period was not tuned. At train time, we present the model with randomly-drawn batches. Consistent with popular experience [36], we find that it is important to use gradient clipping for the RNN, and adaptive gradient clipping worked significantly better than static gradient clipping [37]. In response, we use adaptive gradient clipping for all the deep models. For more information on training and inference performance across the different models, see the Supplementary Methods, and Table 7.

## Rigorous Benchmark Dataset

To benchmark the performance of different models at unaligned protein domain sequence classification and compare deep models to current state of the art models including profile HMMs we use the highly curated Protein families (Pfam) database [11, 23]. The 17929 families of the Pfam 32.0 release are labelled using HMMs that provide broad coverage of the known protein universe; 77.2% of the ∼137 million sequences in UniprotKB have at least one Pfam family annotation, including 74.5% of proteins from reference proteomes [11, 17]. Many domains have functional annotations, although at least 22% of Pfam domains have "unknown function". The HMM for each Pfam family is built from a manually curated family seed alignment, containing between 1 and 4545 protein domain sequences of length 4-2037

amino acids. Some basic statistics about the Pfam family seed sequences can be found in Supplementary Fig. 1.

We split each Pfam family with at least 10 seed sequences randomly into disjoint dev[2] (10%, rounding down to the nearest integer) and test (10%) sets, allocating remaining sequences to the training set (Table 5). Of the 17929 Pfam `seed` families, 4858 families have $< 10$ seed sequences and are only present in the train set. This results in heldout test sequences for 13071 families, where 2819 families have exactly one sequence in each of the test and dev sets. Including families that only exist as training examples makes the task harder because there are more ways each test sequence can be misclassified. Note that we do not expect the HMM-based approach to achieve 100% accuracy because the training data is a subset of the seed data set used in Pfam. For reproducibility, we provide the split Pfam seed dataset for download,[3] and at Kaggle, together with an interactive Jupyter notebook.[4]

For the HMMs, we retain the alignment information from the whole Pfam `seed` for all splits to avoid any artifacts introduced by realignment, and enable optimal performance. During training this provides the HMM with information about the held-out test sequences used to measure performance, meaning that the reported accuracy should be taken as an upper bound. In contrast all alignment information is removed from the data for our deep learning models and for the other baselines.

| | Number of examples | Number of families |
|---|---|---|
| Train | 1086741 | 17929 |
| Dev | 126171 | 13071 |
| Test | 126171 | 13071 |

Table 5: Number of training and testing examples for the randomly split Pfam `seed` data. Note that 16755 families have sequences in the dev and test sets for the Pfam `full` data.

### phmmer

We take the set of unaligned training sequences as a sequence database, and using phmmer from HMMER 3.1b [28] we query each test sequence against this database to find the closest match. Those test sequences that return hits above the default phmmer reporting threshold are then annotated with the label of the training sequence hit with the highest bit score. Out of the 126171 sequences in the test set, 42 did not return a hit using this approach. All training sequences that are not reported as hits by the phmmer function are assumed to have a zero bit score match to that query sequence.

[2] A dev (development) set is a set used to tune hyperparameters that is separate from the test set, to avoid overfitting.

[3] `https://console.cloud.google.com/storage/browser/brain-genomics-public/research/proteins/pfam/random_split`

[4] `https://www.kaggle.com/googleai/pfam-seed-random-split`

Our strategy of working with the Pfam `seed` sequence set circumvents the computationally intensive process of evaluating phmmer and profile HMM performance on the full set of ∼54.5 million Pfam sequences. We did a small amount of tuning to increase the speed of running phmmer so as to give a fair comparison; more information can be found in the supplement.

### k-mer

An alignment-free approach is provided by a k-mer (or n-gram) based model, where each sequence is represented by the set of k-mers that it contains. We train a multi-class logistic regression model on vectors of k-mer counts using the same stochastic gradient descent procedure as used by our deep models (Supplementary Table 1).

### BLASTp

BLASTp [38] is one of the most well known algorithms for searching for similar sequences, and among the current state of the art. It uses an alignment to rank sequences according to their similarity to a target sequence, and from this, a user can impute functional annotation by ascribing known functions of similar sequences. We use BLASTp as a 1-nearest neighbor algorithm by first using `makeblastdb` (version 2.7.1+) with the training data, and then query sequences from that database using `blastp -query`, taking only the top hit. This implementation returns no hit for 259 (0.21%) of the 126171 sequences in the Pfam `seed` test set.

### HMM top pick implementation

Profile HMMs are widely regarded as a state of the art modeling technique for protein sequence classification. We used `hmmbuild` from HMMER 3.1b to construct a profile HMM from the aligned train sequences for each family in Pfam 32.0. We implement a simple top pick HMM strategy (see Methods) to avoid any handicap from the filters built into HMMER 3.1b. To further obtain the best possible profile HMM performance we retain the alignment from the entire Pfam `seed`, avoiding dependence on any particular realignment method. We then use the `hmmsearch` function from HMMER 3.1b to search all 17929 profiles against the set of unaligned test sequences using the default parameter settings.

The scores for each hit are recorded, and the profile with the highest score called as the HMM prediction for that test sequence. To ensure the HMMER 3.1b statistical filters do not hamper performance, we manually turn them off to the extent that at least one hit is reported for each test sequence, and take the top scoring hit. To implement this, for those test sequences with no profile hit after this first pass, we employ a second hmmsearch pass using the '--max' option, which turns off all filters and runs full Forward/Backward inference on every target to increase the sensitivity of the search at a significant cost in speed [39]. In experiments that retain the HMMER 3.1b filters for hmmsearch, we found that 8.5% of test

sequences returned multiple hits above the family specific gathering thresholds that are used by Pfam to regulate family membership. Reporting these results would have resulted in a significantly lower precision score for HMMER than for the deep models, which is why we have chosen instead to remove the statistical filter and report the top hit (Supplementary Fig. 1).

The positive results obtained in the absence of rigorous statistical filters likely reflect the fact that we are working with sequences that were originally classified by Pfam, and so passed the rigorous statistical thresholds set for inclusion in a Pfam family. Those sequences that did not pass these filters, and hence were not included in any Pfam family may well have posed a more significant challenge to our implementation. For this reason we do not recommend that this HMM implementation is used in settings other than working with these benchmark datasets. For Pfam `full` we do not use the HMMs as a baseline because these models were used to label the data, so will achieve 100% accuracy by default. The Pfam `full` dataset has 17772 families overall, and our test and dev sets contain sequences from 16755 families.

## References


[1] Martin Steinegger and Johannes Söding. Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature biotechnology*, 35(11):1026, 2017.

[2] Martin Steinegger and Johannes Söding. Clustering huge protein sequence sets in linear time. *Nature communications*, 9(1):2542, 2018.

[3] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[4] Johannes Söding. Protein homology detection by hmm–hmm comparison. *Bioinformatics*, 21(7):951–960, 2004.

[5] Robert D Finn, Jody Clements, and Sean R Eddy. Hmmer web server: interactive sequence similarity searching. *Nucleic acids research*, 39(suppl_2):W29–W37, 2011.

[6] Morgan N Price, Kelly M Wetmore, R Jordan Waters, Mark Callaghan, Jayashree Ray, Hualan Liu, Jennifer V Kuehl, Ryan A Melnyk, Jacob S Lamson, Yumi Suh, et al. Mutant phenotypes for thousands of bacterial genes of unknown function. *Nature*, page 1, 2018.

[7] Yi-Chien Chang, Zhenjun Hu, John Rachlin, Brian P Anton, Simon Kasif, Richard J Roberts, and Martin Steffen. Combrex-db: an experiment centered database of protein function: knowledge, predictions and knowledge gaps. *Nucleic acids research*, 44(D1): D330–D335, 2015.

[8] Noah M Daniels, Andrew Gallant, Jian Peng, Lenore J Cowen, Michael Baym, and Bonnie Berger. Compressive genomics for protein databases. *Bioinformatics*, 29(13): i283–i290, 2013.

[9] Robert D Finn, Alex Bateman, Jody Clements, Penelope Coggill, Ruth Y Eberhardt, Sean R Eddy, Andreas Heger, Kirstie Hetherington, Liisa Holm, Jaina Mistry, et al. Pfam: the protein families database. *Nucleic acids research*, 42(D1):D222–D230, 2013.

[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553): 436, 2015.

[11] Sara El-Gebali, Jaina Mistry, Alex Bateman, Sean R Eddy, Aurélien Luciani, Simon C Potter, Matloob Qureshi, Lorna J Richardson, Gustavo A Salazar, Alfredo Smart, et al. The pfam protein families database in 2019. *Nucleic acids research*, 47(D1):D427–D432, 2018.

[12] Jie Hou, Badri Adhikari, and Jianlin Cheng. Deepsf: deep convolutional neural network for mapping protein sequences to folds. *Bioinformatics*, 34(8):1295–1303, 2017.

[13] Antonina Andreeva, Dave Howorth, Steven E Brenner, Tim JP Hubbard, Cyrus Chothia, and Alexey G Murzin. Scop database in 2004: refinements integrate structure and sequence family data. *Nucleic acids research*, 32(suppl_1):D226–D229, 2004.

[14] Seokjun Seo, Minsik Oh, Youngjune Park, and Sun Kim. Deepfam: deep learning based alignment-free method for protein family modeling and prediction. *Bioinformatics*, 34 (13):i254–i262, 2018.

[15] Balázs Szalkai and Vince Grolmusz. Near perfect protein multi-label classification with deep neural networks. *Methods*, 132:50–56, 2018.

[16] Balázs Szalkai and Vince Grolmusz. Seclaf: a webserver and deep neural network design tool for hierarchical biological sequence classification. *Bioinformatics*, 34(14):2487–2489, 2018.

[17] UniProt Consortium. Uniprot: the universal protein knowledgebase. *Nucleic acids research*, 45(D1):D158–D169, 2016.

[18] Yu Li, Sheng Wang, Ramzan Umarov, Bingqing Xie, Ming Fan, Lihua Li, and Xin Gao. Deepre: sequence-based enzyme ec number prediction by deep learning. *Bioinformatics*, 34(5):760–769, 2017.

[19] Zhenzhen Zou, Shuye Tian, Xin Gao, and Yu Li. mldeepre: Multi-functional enzyme function prediction with hierarchical multi-label deep learning. *Frontiers in genetics*, 9, 2018.

[20] Ariel S Schwartz, Gregory J Hannum, Zach R Dwiel, Michael E Smoot, Ana R Grant, Jason M Knight, Scott A Becker, Jonathan R Eads, Matthew C LaFave, Harini Eavani, et al. Deep semantic protein representation for annotation, discovery, and engineering. *BioRxiv*, page 365965, 2018.

[21] Da Zhang and Mansur R Kabuka. Protein family classification with multi-layer graph convolutional networks. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 2390–2393. IEEE, 2018.

[22] Xueliang Liu. Deep recurrent neural network for protein function prediction from sequence. *arXiv preprint arXiv:1701.08318*, 2017.

[23] Erik LL Sonnhammer, Sean R Eddy, and Richard Durbin. Pfam: a comprehensive database of protein domain families based on seed alignments. *Proteins: Structure, Function, and Bioinformatics*, 28(3):405–420, 1997.

[24] Steven Henikoff and Jorja G Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.

[25] Andrew Campen, Ryan M Williams, Celeste J Brown, Jingwei Meng, Vladimir N Uversky, and A Keith Dunker. Top-idp-scale: a new amino acid scale measuring propensity for intrinsic disorder. *Protein and peptide letters*, 15(9):956–963, 2008.

[26] Ethan C Alley, Grigory Khimulya, Surojit Biswas, Mohammed AlQuraishi, and George M Church. Unified rational protein engineering with sequence-only deep representation learning. *bioRxiv*, page 589333, 2019.

[27] L Steven Johnson, Sean R Eddy, and Elon Portugaly. Hidden markov model speed heuristic and iterative hmm search procedure. *BMC bioinformatics*, 11(1):431, 2010.

[28] Sean R Eddy. Accelerated profile hmm searches. *PLoS computational biology*, 7(10): e1002195, 2011.

[29] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3319–3328. JMLR. org, 2017.

[30] Brandon Carter, Jonas Mueller, Siddhartha Jain, and David Gifford. What made you do this? understanding black-box decisions with sufficient input subsets. *arXiv preprint arXiv:1810.03805*, 2018.

[31] Brandon Carter, Maxwell Bileschi, Jamie Smith, Theo Sanderson, Drew Bryant, David Belanger, and Lucy Colwell. Critiquing protein family classification models using sufficient input subsets. *In preparation*.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[33] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[34] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[35] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014.

[36] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

[37] Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. End-to-end continuous speech recognition using attention-based recurrent nn: first results. *arXiv preprint arXiv:1412.1602*, 2014.

[38] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[39] Sean Eddy. Hmmer user's guide. biological sequence analysis using profile hidden markov models. 2003.

[40] Robert D Finn, Penelope Coggill, Ruth Y Eberhardt, Sean R Eddy, Jaina Mistry, Alex L Mitchell, Simon C Potter, Marco Punta, Matloob Qureshi, Amaia Sangrador-Vegas, et al. The pfam protein families database: towards a more sustainable future. *Nucleic acids research*, 44(D1):D279–D285, 2015.

[41] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1487–1495. ACM, 2017.

[42] C Nick Pace and J Martin Scholtz. A helix propensity scale based on experimental studies of peptides and proteins. *Biophysical journal*, 75(1):422–427, 1998.