Yamada and Kinoshita *BMC Bioinformatics* (2018) 19:272
https://doi.org/10.1186/s12859-018-2284-1

BMC Bioinformatics

RESEARCH ARTICLE Open Access

De novo profile generation based on sequence context specificity with the long short-term memory network

Kazunori D. Yamada[1,2] and Kengo Kinoshita[1,3,4*]

Abstract

Background: Long short-term memory (LSTM) is one of the most attractive deep learning methods to learn time series or contexts of input data. Increasing studies, including biological sequence analyses in bioinformatics, utilize this architecture. Amino acid sequence profiles are widely used for bioinformatics studies, such as sequence similarity searches, multiple alignments, and evolutionary analyses. Currently, many biological sequences are becoming available, and the rapidly increasing amount of sequence data emphasizes the importance of scalable generators of amino acid sequence profiles.

Results: We employed the LSTM network and developed a novel profile generator to construct profiles without any assumptions, except for input sequence context. Our method could generate better profiles than existing de novo profile generators, including CSBuild and RPS-BLAST, on the basis of profile-sequence similarity search performance with linear calculation costs against input sequence size. In addition, we analyzed the effects of the memory power of LSTM and found that LSTM had high potential power to detect long-range interactions between amino acids, as in the case of beta-strand formation, which has been a difficult problem in protein bioinformatics using sequence information.

Conclusion: We demonstrated the importance of sequence context and the feasibility of LSTM on biological sequence analyses. Our results demonstrated the effectiveness of memories in LSTM and showed that our de novo profile generator, SPBuild, achieved higher performance than that of existing methods for profile prediction of beta-strands, where long-range interactions of amino acids are important and are known to be difficult for the existing window-based prediction methods. Our findings will be useful for the development of other prediction methods related to biological sequences by machine learning methods.

Keywords: Long short-term memory, Deep learning, Neural networks, Sequence context, Similarity search, Protein sequence profile

Background

Amino acid sequence profiles or position-specific scoring matrices (PSSMs) are matrices in which each row contains evolutionary information regarding each site of a sequence. PSSMs have been widely used for bioinformatics studies, including sequence similarity searches, multiple sequence alignments, and evolutionary analyses. In addition, modern sequence-based prediction methods of protein properties by machine learning algorithms often use PSSMs derived from input sequences as input vectors of the prediction. A PSSM is typically constructed from multiple sequence alignment obtained by a similarity search of a query sequence against a huge sequence database such as nr or UniProt [1], and subsequently, the PSSM is refined by iterative database searches. The iteration is a type of machine learning process that improves the quality of profiles gradually. In recent years, HHBlits has been considered the most successful profile generation method [2]. HHBlits generates profiles by iterative searches of huge sequence databases, as in the case of PSI-BLAST [3]; however, HHBlits uses the hidden Markov model (HMM) profile, whereas PSI-BLAST adopts PSSM. To the best of our

* Correspondence: kengo@ecei.tohoku.ac.jp
[1]Graduate School of Information Sciences, Tohoku University, Sendai, Japan
[3]Tohoku Medical Megabank Organization, Tohoku University, Sendai, Japan
Full list of author information is available at the end of the article

knowledge, these methods can produce good profiles on the basis of the performance of similarity searches, but they require an iterative search of a query sequence; therefore, the profile construction time depends on the size of the database. The recent increase in available biological sequences has made it more difficult to construct profiles.

In this context, de novo profile generators such as CSBuild [4, 5] and RPS-BLAST (DELTA-BLAST) [6] have been developed to reduce the cost of profile generation by eliminating the time required for iterative database search, although RPS-BLAST is not exactly a de novo profile generator because it explicitly uses an external profile database. CSBuild internally possesses a 13-mer amino acid profile library, which is a set of sequence profiles obtained by iterative searches of divergent 13-mer sequences. CSBuild searches short profiles against the short profile library for every part of a sequence and subsequently constructs a final profile for the sequence by merging the short profiles. The profile is used as the input data for the similarity search method, CS-BLAST; thus, CSBuild achieves the high performance of a similarity search along with fast computation time. CSBuild can reduce the profile construction time using precalculated short profiles; however, there is no theoretical evidence demonstrating that a PSSM can be constructed by integrating patchworks at the short (13-mer) sequence window. In other words, the previous study assumed that the protein sequences had a short context-specific tendency for the residues. This is also the case with RPS-BLAST, in which a batch of profiles obtained by searches of a query sequence against a precalculated profile library is assembled to construct a final profile.

Recently, neural networks have attracted increasing attention from various research areas, including bioinformatics. Neural networks are computing systems that mimic biological nervous systems of animal brains. Theoretically, if a proper activation function is set to each unit in the middle layer(s) of a network, it can approximate any function [7]. In recent years, neural networks have been vigorously applied to bioinformatics studies. In particular, deep learning algorithms are typically applied to neural networks. For example, several studies have applied deep learning algorithms to predict protein–protein interactions [8, 9], protein structures [10, 11], residue contact maps [12], and backbone angles and solvent accessibilities [13]. The successes of deep learning algorithms have been realized by complex factors, such as recent increases in available data, improvements in the performance of semiconductors, development of optimal activation functions [14], and optimization of gradient descent methods [15]. These various factors have enabled calculations that were thought to be infeasible, and modern deep learning algorithms now not only stack the layers of multilayer perceptrons but also generate various types of inference methods, including stacked autoencoders, recurrent neural networks (RNNs), and convolutional neural networks [14].

The RNN is one of the most promising deep learning methods. More specifically, long short-term memory (LSTM) [16], an RNN, can be a judicious method for learning the time series or context of input vectors. Namely, with LSTM, it may be possible to learn an amino acid sequence context to predict the internal properties of amino acid sequences. The memory of LSTM is experimentally confirmed to be able to continue for more than 1000 time steps, although theoretically, it can continue forever [16]. This memory power may be sufficient to learn features from protein sequences, for which lengths are generally less than 500 amino acids. In addition, compared with window-based prediction methods, we do not need to assume that some protein internal properties, such as secondary structure, steric structure, or evolutionary information, are formed in some lengths of amino acid sequences, as in the case of CSBuild, which assumes 13-mers. LSTM can even learn such optimal lengths of context automatically throughout learning. This characteristic of LSTM is thought to be more suitable for protein internal property predictions. Indeed, several machine learning–based prediction methods utilizing the LSTM network for protein property prediction have been successful applied [13, 17, 18].

In this study, we attempted to develop a de novo profile generator that mimicked the ability of the existing highest performance profile generation method, HHBlits, using an LSTM network, expecting our generator to be able to include the ability to input whole protein sequences. In addition, we analyzed the importance of sequence context in the prediction and performance of LSTM to solve specific biological problems through our computational experiments.

Methods

Learning dataset

We conducted iterative searches using HHBlits version 2.0.15 with the default iteration library provided by the HHBlits developer and generated profiles of the sequences in Pfam version 29.0 [19], where the sequences were clustered by kClust version 1.0 [20] and the maximum percent identity for all pairs of sequences was less than 40% (Pfam40). Because we used the SCOP20 test dataset as a benchmark dataset for the performance of profile generators (see below), we excluded highly similar sequences with any sequences in the SCOP20 test dataset from the Pfam40 dataset using gapped BLAST (blastpgp) searches prior to the iterative search, where we considered retrieved sequences with an e-value of less than 10^{-10} as the highly similar sequences. The number of HHBlits iterations was set to three. Although

HHBlits produces HMM profiles, we converted these profiles to PSSMs by extracting amino acid emission frequencies of match states. Finally, we set the generated profiles as target vectors and its corresponding sequences as input vectors in learning steps. Namely, in our learning scheme, each instance included an N dimension vector (sequence) as an input vector and a $20 \times N$ dimension vector (profile) as a target vector, where N represents sequence length and 20 is the number of types of amino acid residues.

Learning network

We designed a network with an LSTM layer, as shown in Fig. 1a. In the figure, the numbers at the bottom of the panel represent the number of dimensions of the vectors at each layer. In the learning steps, each amino residue (one character = one-dimension integer vector) in the input sequence was converted to a 400-dimension floating vector by the word embedding method. Word embedding is a technique used to increase the expressiveness of a learning network and generally improves the learning performance of the network [21, 22]. One of the advantages of the encoding method over the normal one-hot encoding method, where an amino acid residue is encoded by a 20-dimension sparse integer vector comprising a single one and 19 zeros, is that the encoding method uses a floating non-zero value in the vector. Since the value of the next layer is calculated by multiplying a vector on the present layer with a parameter matrix of the network, the sparse vector with many zeros cannot effectively use the parameters, because multiplication including zero generates only zero value (less information). In addition, increasing the dimension of the first layer using the encoding method will have a good effect on the learning network, because a moderately wider first layer can keep the next layer narrower while yielding the same magnitude of parameters. The narrower layer is advantageous over a wider layer in that it can reduce overfitting. Generally, a narrow-deeper network has higher learning performance than a wide-shallower network [23–25]. After the word embedding process, the input vectors were processed by an LSTM layer followed by a fully connected layer. The dimension of the fully connected layer was set to 20 to correspond to the number of types of amino acid residues. The output of the network was set to a solution of the softmax function of the immediately anterior layer. Because the summation of a solution of the softmax function is one, we can interpret the values as a probability, i.e., the amino acid probability on each site in the study. With the probability vectors, we can reproduce PSSM. We set the unit size of each gate of the LSTM unit to 3200. As a cost function, we used the root mean square error between an output of the network and a target vector. As an optimizer of the gradient descent method, Adam was used [15]. As an LSTM unit, we utilized an extended LSTM with a forget gate [26], as shown in Fig. 1b. In Fig. 1b, the top, middle, and bottom sigmoid gates represented the input, forget, and output gates, respectively. LSTM imitates the mechanism of an animal brain using these gates. In addition, by storing the previous computation results in the memory cell, $\boldsymbol{h}$, and using it at the next computation, LSTM can memorize a series of previous incidents, thus gaining context. For regularization aimed at reducing the risk of overfitting, we used a dropout method against weights between an input layer and an LSTM layer with a drop ratio of 0.5, and based on the ratio, neurons were stochastically inactivated. We observed learning and validation curves to avoid overfitting and stopped learning steps at 5000 epochs. Because we could not deploy whole sequence data into the memory space at one time in our computational environment, we randomly selected 40,000 sequences (about 1/40th of all sequences) and learned them as a single epoch. Therefore, an epoch in this study was about 40 times the typical epoch. Here, epoch means the number of parameters updated during the inference, i.e., the progress of learning.

As a framework to implement the learning network, we used Chainer version 1.15.0.1 (Preferred Networks) with CUDA and cuDNN version 6.5 (NVIDIA), and the calculations were performed by a server with Tesla K20m (NVIDIA) at the NIG supercomputer at ROIS National Institute of Genetics in Japan.

Benchmark of the performance of similarity searches

Performances of profile generators were evaluated based on the results of similarity searches with their generated profiles. As representatives of existing methods of rapid profile generators, we compared our method with CSBuild version 2.2.3 and RPS-BLAST version 2.2.30+. As a test dataset, the SCOP20 test dataset was used, as in the original paper for CSBuild [5], which consists of 5819 sequences with protein structural information; the maximum percent identity of the sequences in the dataset was less than 20%. In addition to the dataset, we constructed another test dataset as a SCOP20 strict-test dataset. To construct the dataset, we excluded homologous sequences with any sequence in the Pfam40 learning dataset from the SCOP20 test dataset using blastpgp searches with an e-value of less than 10^{-5} as the threshold of homologous hits. As a result, the SCOP20 strict-test dataset contained 1104 sequences. As a profile library for CSBuild, the data from the discriminative model of CSBuild (K4000.crf) were used. For RPS-BLAST, we excluded all highly similar sequences with any sequence in the SCOP20 test dataset from the conserved domain database for DELTA-BLAST version 3.12 by the same method as that used to make the Pfam40 learning dataset.

To eliminate any biases of alignment algorithms, all profiles in this study were converted to the PSI-BLAST

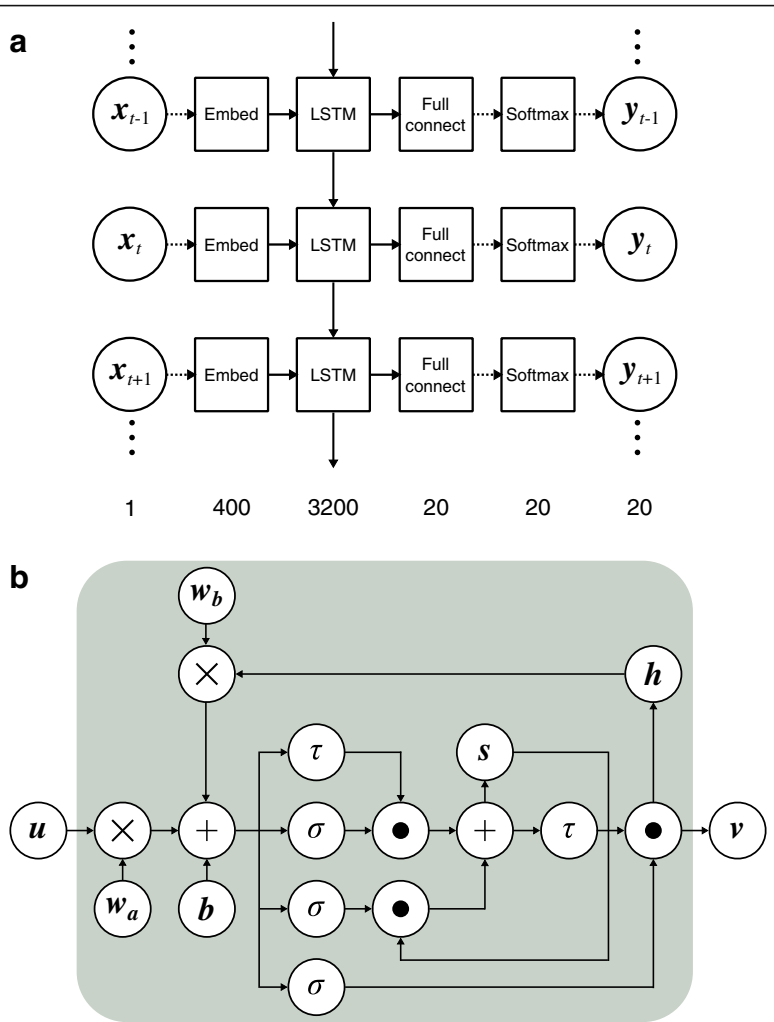

Fig. 1 Network of learning. **a** Overview of the designed network in this study. Here, ***x***, ***y***, and *t* represent an input vector, an output vector, and a position of an amino acid sequence. In the squares, "Embed," "Full connect," and "Softmax" stand for a word embedding operation, a fully connected network, and a softmax function layer, respectively. The solid and broken arrows represent a matrix operation and an array operation, respectively. The numbers at the bottom of panel (**a**) stand for a dimension of vectors of each layer. **b** Description of LSTM layer. Here, ***u***, ***v***, ***h***, ***s***, ×, +, dot, τ, σ, $\boldsymbol{w}_{\sigma}$, $\boldsymbol{w}_{b}$, and ***b*** stand for an input vector to an LSTM unit, an output vector from an LSTM unit, a previous input vector, a unit for constant error, multiplication of matrices, summation of matrices, a Hadamard product calculation, a hyperbolic tangent, a sigmoid function, a weight matrix to be learned, another weight matrix, and a bias vector

readable format and used as input files in a PSI-BLAST search. As an application of PSI-BLAST, we used blastpgp version 2.2.26 for CSBuild, since CSBuild outputs blastpgp-readable profile files. For the other methods, psiblast version 2.2.30+ was used. There were no significant differences in sensitivity or similarity searchers between these two versions of PSI-BLAST (data not shown). The results of the similarity searches were sorted according to their statistical significance in descending order. Each hit was labeled as a true positive, false positive, or unknown based on the evaluation ruleset for SCOP 1.75 benchmarks (http://supfam.cs.bris.ac.uk/SUPERFAMILY/ruleset_1.75.html) [27]. Further, the number of true positives and false positives was normalized by weighting them with the number of members in each SCOP *superfamily* to negate bias derived from the size of each SCOP *superfamily*. With this information, we described the receiver operating characteristic (ROC) curves and evaluated the performance [28]. As an evaluation criterion, we used partial area under the ROC curve (pAUC), which is the AUC until one false positive is detected for each query on average. In our case, the pAUC was equivalent to AUC until 1564 false positives in total were detected, because we weighted detected false positives by the size of each SCOP *superfamily*, and the number of *superfamilies* in our test dataset is 1564.

The profile generation time was benchmarked on an Intel(R) Xeon(R) CPU E5–2680 v2 @2.80 GHz with 64 GB RAM using a single thread.

Results and discussion

Training a predictor with LSTM

In this study, we assumed profiles generated by HHBlits as ideal profiles and used these as target profiles in training steps. We then attempted to generate profiles as similar to the HHBlits profiles as possible with a predictor using LSTM. The performances of similarity searches with the profiles generated by HHBlits were better than those of the other methods [2].

Initially, we selected amino acid sequences with lengths of 50–1000 in Pfam40. The sequences did not contain any irregular amino acid characters such as B, Z, J, U, O, or X. As a result, we obtained 1,602,338 sequences and calculated their profiles using HHBlits for each sequence. We also included 1329 sequences derived from the SCOP20 learning dataset [5] to the final learning dataset in order to analyze the developed method further (see "Performance comparisons" in the section.). With this learning dataset, we trained the predictor shown in Fig. 1a. In order to check whether the predictor overfit the training dataset, we used 20,000 randomly extracted instances as a validation dataset and monitored the training and validation curves. The number of mini-batches was set to 200, and each amino acid was converted to a 400-dimension floating vector by the word embedding method, as described in the methods section. For each sequence, the starting site of learning was not confined to the N-terminal but was selected at random to avoid overfitting of the predictor to the specific site. The training and validation curves did not deviate from each other, confirming the absence of overfitting, and stopped learning at 5000 epochs (Additional file 1: Figure S1). Even using the GPU machine, the completion of our calculations required almost two months.

Using the obtained parameters (weight matrices and bias vectors through the learning), we constructed a novel de novo profile predictor, which we called Synthetic Profile Builder (SPBuild). Our profile generator can be downloaded from http://yamada-kd.com/product/spbuild.html.

Performance comparisons

First, we compared the performance of the similarity searches of the profile generators. The profiles for all sequences in the SCOP20 test dataset were generated by each method, and all-against-all comparisons of the test dataset by PSI-BLAST with the obtained profiles were conducted. As profile generators, we evaluated the de novo profile generators CSBuild and RPS-BLAST, in addition to SPBuild. We also added the performance of PSI-BLAST without iterations (= blastpgp) as a representative sequence–sequence-based alignment method for reference. In addition, HHBlits was further compared as another reference, and the results are shown in Additional file 1: Figure S2.

As shown in Fig. 2a, CSBuild and RPS-BLAST were clearly superior to the sequence–sequence-based alignment method, blastpgp. Furthermore, SPBuild showed better performance than those of these methods. When performance was evaluated by the pAUC values (Fig. 2b), the values of our method, CSBuild, and RPS-BLAST were 0.217, 0.140, and 0.174, respectively. Notably, the performance of our method (0.217) did not reach that of HHBlits (Additional file 1: Figure S2a, pAUC = 0.451), even though we trained our predictor with outputs of HHBlits, indicating that SPBuild was not completely able to mimic the ability of HHBlits. This tendency was also true for another benchmark result, where we evaluated the performance of

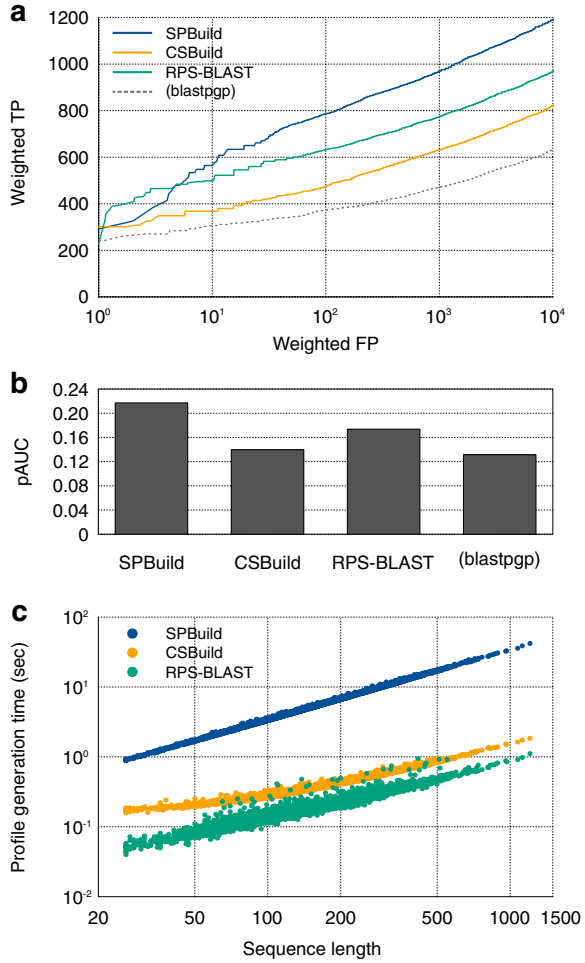

Fig. 2 Performance comparisons of (**a**, **b**) similarity searches and (**c**) calculation time. **a** ROC curves of SPBuild and other methods. Here, the performance of blastpgp was added for a reference. **b** The pAUC values of SPBuild, CSBuild, RPS-BLAST, and blastpgp. **c** The scatterplot of the profile generation time for each method on the SCOP20 test dataset

SPBuild and HHBlits on the SCOP20 learning dataset instead of the test dataset (Additional file 1: Figure S2b). Our findings were surprising because the SCOP20 learning dataset was a part of the learning dataset for the construction of the predictor with LSTM, and the performance of our predictor should reach that of HHBlits. One possible reason for the observation is that LSTM may not have worked properly on our learning scheme. To examine this possibility, we performed another learning method to examine the performance of LSTM itself with our learning scheme, where we trained a predictor with only the SCOP20 learning dataset and let the predictor overfit the dataset. As a result, the performance of the predictor was almost the same as that of HHBlits, as expected (Additional file 1: Figure S2c). This result indicated that LSTM could precisely learn input sequence properties and output correct PSSMs, but that the performance of the predictor was worse than that of SPBuild with proper learning due to the overfitting of the predictor to the learning dataset (Additional file 1: Figure S2d). In short, these results suggested that LSTM worked correctly, and that relationship between performance and overfitting was a simple trade-off. Therefore, we concluded that SPBuild could be trained moderately and pertinently without conflict under our learning dataset and hyperparameters.

Next, we evaluated the profile generation time of each method. Table 1 shows the mean computation time of profile generation using the SCOP20 test dataset. SPBuild was found to be almost 20 times faster than HHBlits, although CSBuild and RPS-BLAST were still faster than SPBuild. However, we think the most important property of a sequence handling method in the big data era is scalability to the data, namely, time complexity of the method against the input sequence length. Theoretically, the time complexity of our method would be linear compared with the input sequence length, similar to CSBuild and RPS-BLAST. To clarify this point, we plotted profile generation times (seconds) versus input sequence lengths (N), as shown in Fig. 2c. When the instances were fitted to a line, the determination coefficient was 0.998, and the slope of the line was 1.00. This result indicated that the time complexity of our method was $O(N)$. Notably, the slopes of CSBuild and RPS-BLAST appeared to be less than 1.0 in the figure; however, errors in the experiments or other factors in the implementation of these programs may have caused this because the costs of these calculations must be higher than that of $O(N)$. Actually, if we conducted a similar experiment using simulation-sequence data with longer sequences, the slopes of CSBuild and RPS-BLAST were about 1.01 and 0.93 and the profile generation time was almost linear against sequence length (Additional file 1: Figure S3). Although our method required much time to compute large matrix calculations in the neural network layers and was therefore slower than CSBuild and RPS-BLAST with the currently used sequence database, our method had linear scalability against the number of input sites or sequence length and the number of input sequences. Although the time complexity of de novo profile generators, including SPBuild, is $O(N)$, that of HHBlits and other iterative methods is also linear to the length of query sequences. The difference in the methods lies in the requirement for iterative search in a large database. The de novo profile generators achieved faster profile generation time because they succeeded in eliminating the cost of searching the large database.

Table 1 Comparison of profile generation times

	Mean	SD
SPBuild	5.99	3.83
CSBuild	0.390	0.161
RPS-BLAST	0.208	0.102
HHBlits	120	105

Means and standard deviations (SDs) of profile generation times (s) against 5819 sequences in the SCOP20 test dataset

Memory power of LSTM in our problem

We also examined the memory power of LSTM in our problem to determine the feasibility of the LSTM approach for sequence-based predictions. For this, we considered the reset time lengths of memory cells ($\boldsymbol{h}$ in Fig. 1b) at sequence lengths of 5, 10, 20, 30, 50, 100, 200, and 300 and for full-length sequences (= SPBuild). We then benchmarked the performances of similarity searches with the SCOP20 test dataset. The memory reset time length was directly linked to the memory power of the predictors, and a predictor with a memory reset time length of 5, for example, generated profiles based on information from the previous five sites, including the current site. As a result, the performance of similarity searches clearly changed as the memory power decreased (Fig. 3a).

We also checked the performance of CSBuild with the same plot (Fig. 3a). As described above, CSBuild constructs profiles by merging 13-mer short profiles; thus, we imagined that its performance would be similar to that of the LSTM profile predictors with low memory power. However, we found that the performance of CSBuild was located in the middle, between memory powers of 30 and 50 for the LSTM predictors. We are not sure why this happened, but it might be because the sensitivities (corresponding to the vertical axis of Fig. 3a) of LSTM predictors were worse than expected or because of the excellence of CSBuild implementations.

To improve our understanding of the generated profiles by SPBuild, we evaluated the mean prediction accuracy (cosine similarity between the output vector $\boldsymbol{y}$ and the target vector) of SPBuild for each position of a residue on whole input sequences and observed that

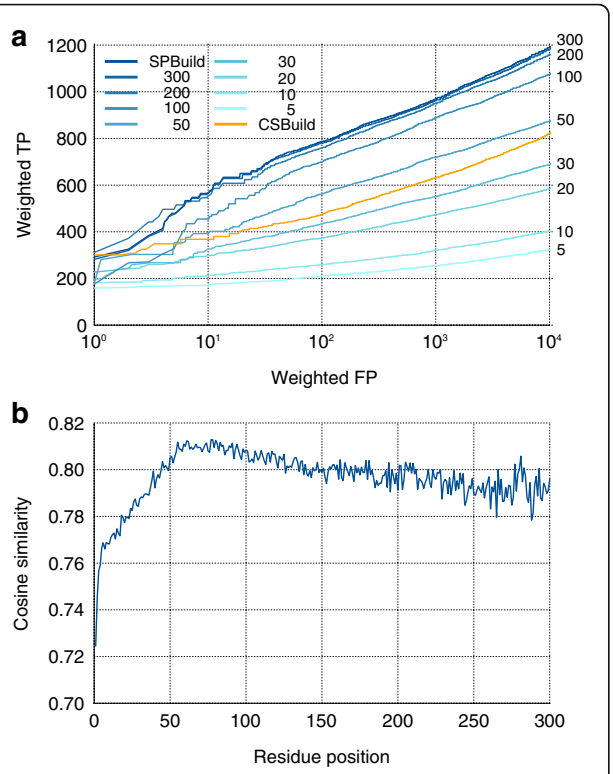

Fig. 3 Effects of memory power of LSTM on predictors. **a** Comparison of profile generators with various reset lengths of memory on LSTM. The benchmark dataset was the SCOP20 test dataset. The reset time of SPBuild corresponded to the input sequence length. **b** Mean cosine similarity between output vectors of SPBuild and target vectors as a function of the position of residues in input sequences of the SCOP20 test dataset

there was a clear transition in the plot (Fig. 3b). The prediction accuracy of the initial portion (~ 50) was worse than those of the other parts. This lower performance could be caused by the nature of LSTM. LSTM initializes the internal state of memory (***h***) by a null vector, which does not reflect any features of the learning dataset; thus, the prediction would be not stable until LSTM memorizes and stores a certain level of context information into memory. In our case, the level of context information was 50–60 residues. In addition, the decrease in accuracy in the last part (~ 200) was derived from the nature of our learning dataset; the mean length of SCOP20 was about 154, and SPBuild may be able to be optimized for the average length. This consideration was consistent with the observation that improvement of the performance with memory power of 200 and 300 decreased compared with smaller memory power lengths (Fig. 3a). On the basis of the observations that the prediction confidence of the N-terminal region was not good, we think that it might be possible to improve the performance of SPBuild by combining prediction results from both N-terminal and C-terminal directions. Although we did not implement this feature because the learning process took lots of time, this will be a future direction for further improvements.

In conclusion, these results suggested that substantially long length context—ideally speaking, the context of the sequence length of at least more than 50—would be required to predict precise profiles. Protein primary and secondary structures, including solvent accessibility and contact number, must be restricted by not only their sequentially local interactions but also the three-dimensional interactions of the residues inside their protein steric structures, which are formed by spatially complex remote interactions of amino acid residues. For example, hydrophobic residues tend to be located inside the protein structure and aliphatic residues tend to be located on the β-sheet [29–31]. Our findings reflect the influence of remote relationships stemming from the steric structure on sequence context. In other words, LSTM will be a powerful predictor for divergent features of proteins, if appropriate memory power length is used. Indeed, other sequence-based predictors using LSTM have achieved successful outcomes and have shown the high feasibility of LSTM [13, 17, 18].

Long-range interactions and memory lengths

As shown in Fig. 4a, we calculated the pAUC values of SPBuild relative to those of CSBuild and RPS-BLAST for each SCOP class. The values were calculated by dividing the pAUC value of SPBuild by that of each method, which indicated how the sensitivity of SPBuild was better than those of the existing methods for each SCOP class. Actual pAUC values are shown in Additional file 1: Table S1. Notably, the performance of SPBuild was 2.00- and 1.49-fold higher than those of CSBuild and RPS-BLAST for SCOP class b, respectively. SCOP class b consists of β proteins. Generally, β-strands are constructed by remote interactions between residues when compared with α-helices. Secondary structure predictors with a window-based method developed by machine learning methods tend to show poorer performance in β-regions than in α-regions. The main reason for this weakness is related to the long-range interactions in β structures, which may not be properly handled by the limited lengths of sequence windows [32, 33]. This tendency may also be observed with the profile predictors. CSBuild constructs final profiles by assembling short window-based profiles, and RPS-BLAST also combines many subjected profiles obtained by local similarity searches against profile libraries. The actual mean length of the profiles evaluated by RPS-BLAST with three iterations (default) on the SCOP20 test dataset was 77, which was relatively longer than that of CSBuild but still shorter than the typical length of a protein. However, our method can theoretically memorize whole-length amino acid sequences and can take the remote relationship into consideration to generate profiles.

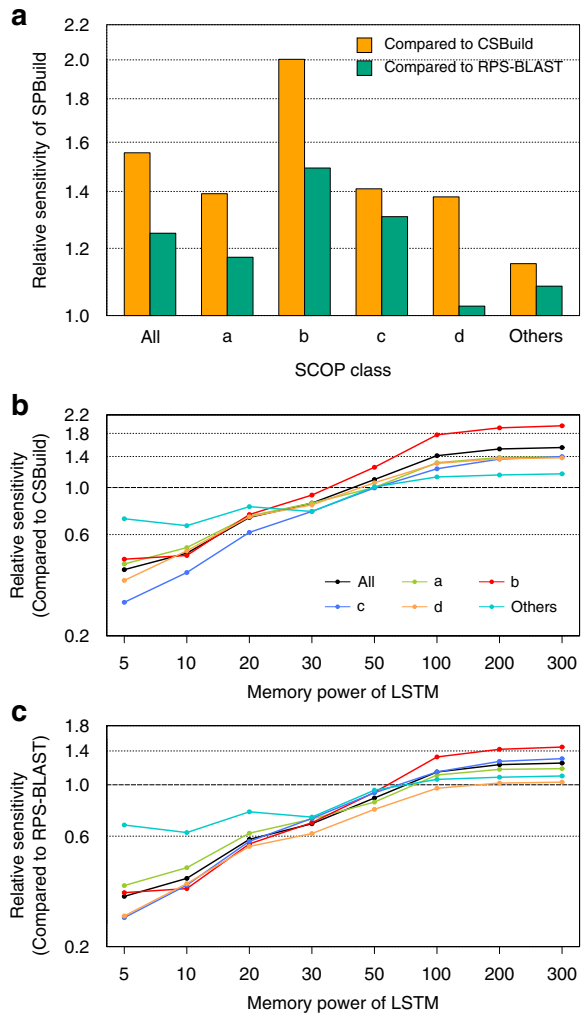

Fig. 4 Relative sensitivity of SPBuild against that of existing methods on the test dataset. **a** The relative sensitivity of SPBuild against existing methods was calculated by dividing the pAUC of SPBuild by that of each method. Here, the label "others" includes SCOP classes e, f, and g. **b** The relative sensitivity of the profile generator with various memory powers of LSTM against CSBuild. **c** The relative sensitivity of the profile generator with various memory powers of LSTM against RPS-BLAST

To confirm the relationship between memory power length and structural categories, we calculated relative sensitivities for different reset time lengths (Fig. 4b and c). As a result, the performance improvements in the b category were much better than those of other categories, indicating that memory power was the most important factor for encoding long-range interactions, such as β structures.

Limitations of SPBuild

As described, our method could generate profiles faster than HHBlits, and it demonstrated superior performance to those of CSBuild and RPS-BLAST, particularly for β-region prediction, possibly due to the memory effects of LSTM. However, there are still some limitations to this method.

One of the limitations of SPBuild is the profile generation time, although the time complexity is linear against input sequence length. SPBuild used huge numbers of parameters, particularly for the LSTM layer, to calculate the final profile prediction. Although we set the size of the parameters to the current scale to maximize the final performance of SPBuild, we may be able to reduce the size and improve the calculation time if we are able to find more efficient network structures to learn amino acid context. In other words, to resolve the problem, exhaustive optimization of the hyperparameters of LSTM and/or development of novel network structures will be required.

For the construction of the Pfam40 learning dataset, we excluded highly similar sequences with any sequence in the SCOP20 test dataset from the original Pfam40 dataset by blastpgp search having e-value $< 10^{-10}$. It should be noted that the threshold is rather strict to eliminate homologous sequences. In the context of machine learning, the independence of the test and learning dataset is quite important to avoid overtraining, and thus, the same data among the datasets should be eliminated, but similar data are usually retained for better learning. Generally, a test dataset must follow the same probability distribution as that of the learning dataset [34, 35]. In other words, the existence of similar data among a learning and test set is an essential point for supervised learning, and prediction based on supervised learning will fail if no similar data are available among the learning and test dataset. The similar information will be a question of degree, and in our case, better learning would require a homologous relationship in both the learning and test dataset.

Meanwhile, however, in the context of biological sequence analysis, homologous or similar sequences will be conceptual problems. From the viewpoint of machine learning, homologous sequences should not be removed, but conventional approaches of biological sequence analyses usually remove the homologous sequences [36–38]. For further considerations, we set a moderate e-value threshold of 10^{-5} aiming to exclude homologous sequences in the Pfam40 learning dataset from the SCOP20 test dataset, and we made another test dataset, a SCOP20 strict-test dataset. According to benchmark results with the dataset (Fig. 5), the search sensitivities of de novo profile generators including SPBuild were much lower than that of HHBlits, and our method was worse than blastpgp, which is a sequence–sequence-based method. These results will be quite interesting to understand profile generation with machine learning approaches and indicate that machine learning approaches would not be effective at all if homologous sequences are excluded, as

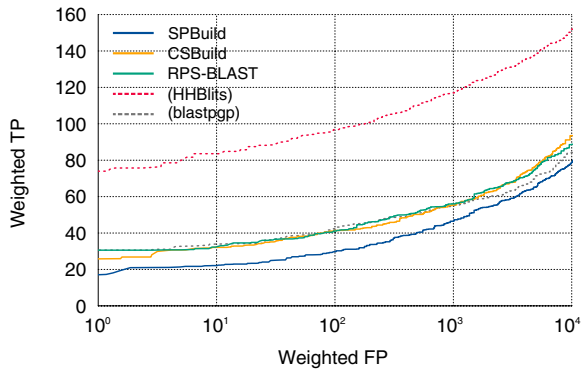

Fig. 5 Performance comparisons of similarity searches on SCOP20 strict-test dataset. ROC curves of SPBuild and other methods. The performances of HHBlits (three iterations) and blastpgp were added for a reference

conventional sequence analyses methods are doing. On the other hand, the worse performance of SPBuild might be improved to at least the same level as that of blastpgp by introducing a bailout method, which is a popular approach in machine learning, where profiles are generated from the background frequency of amino acid substitution matrices like BLOSUM [39] or MIQS [40] when the confidences of profile generation are not enough. That kind of bailout is internally implemented by BLAST series, but we did not use it in the current implementations, and thus, it can be a future direction for further improvements. In practical use, our predictor will not be able to find completely novel sequences that do not share any homologous relationships with the sequences in a training dataset, despite the training with all the available sequence data in the world. Thus, our predictor will be a profile generator capable of generating profiles of existing or similar sequences rapidly, and its concept is similar to that of RPS-BLAST.

The performance of iteration search with profiles made by de novo profile generators would be another interesting point for users. To check the performance of iteration searches, we calculated ROC curves for SPBuild, CSBuild, and RPS-BLAST and found that the performance differences diminished as the number of iterations increased (Additional file 1: Figure S4). The result suggested that the performance of the initial search or qualities of profiles would be of meager importance for the final results in iterative searches if a sufficient number of iterations was used. The reason for this result is unclear; however, we believe that the number of homologous sequences in the sequence space is not infinite and that almost all homologous sequences can be detected by using modestly good profiles if a large number of iterations are used. Considering the sensitivity of profile sequence–based similarity searches, our method may not be too attractive; however, there are many other uses for profiles. For example, profile–profile similarity searches, where profiles are generated by iterative searches of whole datasets, will be candidates for the application of our approach. The bottleneck of profile–profile searches may be easily resolved with the rapid profile generator. In addition, profiles are often used to encode amino acids into input vectors in other machine learning methods. Machine learning methods generally require large sets of learning data, and currently, long-time iterative searches should be avoided because the calculation time increases depending on the learning data size. In such cases, higher speeds and accurate profile generators will be quite useful.

Conclusions

In this study, we developed a novel de novo generator of PSSMs using a deep learning algorithm, the LSTM network. Our method, SPBuild, improved the performance of homology detection with a more rapid computation time than that of existing de novo generators. However, our goal was not to just provide an alternative method for profile generators but also to elucidate the importance of sequence context and the feasibility of LSTM for overcoming the sequence-specific problem. Our analyses demonstrated the effectiveness of memories in LSTM and showed that SPBuild achieved higher performance, particularly for β-region profile generation, which was difficult to predict by window-based prediction methods. This performance could be explained by the fact that our method utilized the LSTM network, which could capture remote relationships in sequences. Moreover, further analyses suggested that substantially long context was required for correct profile generation. We also reconfirmed several limitations of deep learning on our problems. For example, the deep architecture to realize higher performance required considerable computation time, and the intensive elimination of homologous information between the learning and test dataset might make the inference by deep learning impossible. These findings may be useful for the development of other prediction methods.

We have not developed a profile generator with a performance superior to that of HHBlits, and this was not our objective either. Actually, we adopted the supervised learning method, where the predictor basically would not be able to superior to the teacher. However, as in the case of AlphaGo Zero [41], state-of-the-art learning methods such as reinforcement learning may enable us to develop an alternative method for HHBlits.

Profiles are the most fundamental data structures and are used for various sequence analyses in bioinformatics studies. Using SPBuild, the performance of sophisticated comparison algorithms, such as profile–profile comparison methods and multiple sequence alignment, can be further improved. In addition, profiles generated by SPBuild can be useful as input vectors for other machine-based meta-predictors of protein properties.

Additional file

Additional file 1: Figure S1. Learning curves of the LSTM, **Figure S2.** ROC curves of similarity search for the target (HHBlits) and predictors, **Figure S3.** Comparison of profile generation time with simulation data, **Figure S4.** ROC curves of the similarity search for each iterative method, **Table S1.** Comparison of pAUC values for SCOP classes for SCOP20 test datasets. (PDF 857 kb)

Abbreviations

HMM: Hidden Markov model; LSTM: Long short-term memory; pAUC: Partial area under the ROC curve; PSSM: Position-specific scoring matrix; RNN: Recurrent neural network; ROC: Receiver operating characteristic

Acknowledgements

We are grateful to Kentaro Tomii and Toshiyuki Oda for constructive discussions. Computations were partially performed on the NIG supercomputer at ROIS National Institute of Genetics and the supercomputer system Shirokane at Human Genome Center, Institute of Medical Science, University of Tokyo.

Funding

This work was supported in part by the Top Global University Project from the Ministry of Education, Culture, Sports, Science, and Technology of Japan (MEXT), KAKENHI from the Japan Society for the Promotion of Science (JSPS) under Grant Number 18K18143 and Platform Project for Supporting in Drug Discovery and Life Science Research (Basis for Supporting Innovative Drug Discovery and Life Science Research (BINDS)) from AMED under Grant Number JP18am0101067. The funding bodies did not play any role in the design of the study nor collection, analysis, nor interpretation of data nor in writing the manuscript.

Availability of data and materials

The source code of SPBuild is available at http://yamada-kd.com/product/spbuild.html.

Authors' contributions

KDY conducted the computational experiments and wrote the manuscript. KK supervised the study and wrote the manuscript. Both authors have read and approved the final manuscript.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Author details

[1]Graduate School of Information Sciences, Tohoku University, Sendai, Japan. [2]Artificial Intelligence Research Center, National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan. [3]Tohoku Medical Megabank Organization, Tohoku University, Sendai, Japan. [4]Institute of Development, Aging, and Cancer, Tohoku University, Sendai, Japan.

Received: 17 May 2018 Accepted: 11 July 2018
Published online: 18 July 2018

References

1. Ncbi-Resource-Coordinators. Database resources of the National Center for biotechnology information. Nucleic Acids Res. 2017;45(D1):D12–7.
2. Remmert M, Biegert A, Hauser A, Soding J. HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment. Nat Methods. 2011; 9(2):173–5.
3. Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res. 1997;25(17):3389–402.
4. Biegert A, Soding J. Sequence context-specific profiles for homology searching. Proc Natl Acad Sci U S A. 2009;106(10):3770–5.
5. Angermuller C, Biegert A, Soding J. Discriminative modelling of context-specific amino acid substitution probabilities. Bioinformatics. 2012;28(24):3240–7.
6. Boratyn GM, Schaffer AA, Agarwala R, Altschul SF, Lipman DJ, Madden TL. Domain enhanced lookup time accelerated BLAST. Biol Direct. 2012;7:12.
7. Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal Approximators. Neural Netw. 1989;2(5):359–66.
8. Sun T, Zhou B, Lai L, Pei J. Sequence-based prediction of protein protein interaction using a deep-learning algorithm. BMC Bioinformatics. 2017; 18(1):277.
9. Du X, Sun S, Hu C, Yao Y, Yan Y, Zhang Y. DeepPPI: boosting prediction of protein-protein interactions with deep neural networks. J Chem Inf Model. 2017;57(6):1499–510.
10. Wang S, Peng J, Ma J, Xu J. Protein secondary structure prediction using deep convolutional neural fields. Sci Rep. 2016;6:18962.
11. Spencer M, Eickholt J, Cheng J. A deep learning network approach to ab initio protein secondary structure prediction. IEEE/ACM Trans Comput Biol Bioinform. 2015;12(1):103–12.
12. Di Lena P, Nagata K, Baldi P. Deep architectures for protein contact map prediction. Bioinformatics. 2012;28(19):2449–57.
13. Heffernan R, Yang Y, Paliwal K, Zhou Y. Capturing non-local interactions by long short term memory bidirectional recurrent neural networks for improving prediction of protein secondary structure, backbone angles, contact numbers, and solvent accessibility. Bioinformatics 2017;33(18):2842–9.
14. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521(7553):436–44.
15. Kingma D, Ba J. Adam: a method for stochastic optimization. In: arXiv preprint arXiv:14126980; 2014.
16. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1735–80.
17. Hanson J, Yang Y, Paliwal K, Zhou Y. Improving protein disorder prediction by deep bidirectional long short-term memory recurrent neural networks. Bioinformatics. 2017;33(5):685–92.
18. Kim L, Harer J, Rangamani A, Moran J, Parks PD, Widge A, Eskandar E, Dougherty D, Chin SP. Predicting local field potentials with recurrent neural networks. Conf Proc IEEE Eng Med Biol Soc. 2016;2016:808–11.
19. Finn RD, Coggill P, Eberhardt RY, Eddy SR, Mistry J, Mitchell AL, Potter SC, Punta M, Qureshi M, Sangrador-Vegas A, et al. The Pfam protein families database: towards a more sustainable future. Nucleic Acids Res. 2016;44(D1):D279–85.
20. Hauser M, Mayer CE, Soding J. kClust: fast and sensitive clustering of large protein sequence databases. BMC Bioinformatics. 2013;14:248.
21. Habibi M, Weber L, Neves M, Wiegandt DL, Leser U. Deep learning with word embeddings improves biomedical named entity recognition. Bioinformatics. 2017;33(14):I37–48.
22. Asgari E, Mofrad MRK. Continuous distributed representation of biological sequences for deep proteomics and genomics. PLoS One. 2015;10(11): 0141287.
23. Yu D, Seltzer ML, Li J, Huang J-T, Seide F. Feature learning in deep neural networks-studies on speech recognition tasks. In: arXiv preprint arXiv: 13013605; 2013.
24. Ciregan D, Meier U, Schmidhuber J: Multi-column deep neural networks for image classification. In: Computer vision and pattern recognition (CVPR), 2012 IEEE conference on: 2012. IEEE: 3642–3649.
25. Ciresan DC, Meier U, Masci J, Maria Gambardella L, Schmidhuber J: Flexible, high performance convolutional neural networks for image classification. In: IJCAI proceedings-international joint conference on artificial intelligence: 2011. Barcelona, Spain: 1237.
26. Gers FA, Schmidhuber J, Cummins F. Learning to forget: continual prediction with LSTM. Neural Comput. 2000;12(10):2451–71.
27. Gough J, Karplus K, Hughey R, Chothia C. Assignment of homology to genome sequences using a library of hidden Markov models that represent all proteins of known structure. J Mol Biol. 2001;313(4):903–19.
28. Gribskov M, Robinson NL. Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching. Comput Chem. 1996;20(1):25–33.
29. Rose GD, Geselowitz AR, Lesser GJ, Lee RH, Zehfus MH. Hydrophobicity of amino acid residues in globular proteins. Science. 1985;229(4716):834–8.
30. Chou PY, Fasman GD. Prediction of protein conformation. Biochemistry. 1974;13(2):222–45.

31. Shirota M, Ishida T, Kinoshita K. Effects of surface-to-volume ratio of proteins on hydrophilic residues: decrease in occurrence and increase in buried fraction. Protein Sci. 2008;17(9):1596–602.
32. Bradley P, Baker D. Improved beta-protein structure prediction by multilevel optimization of nonlocal strand pairings and local backbone conformation. Proteins. 2006;65(4):922–9.
33. Cheng J, Baldi P. Three-stage prediction of protein beta-sheets by neural networks, alignments and graph algorithms. Bioinformatics. 2005;21(Suppl 1):i75–84.
34. Bishop CM. Pattern recognition and machine learning. New York: Springer; 2006.
35. Goodfellow I, Bengio Y, Courville Y: Deep learning: MIT Press; 2016.
36. Soding J, Remmert M. Protein sequence comparison and fold recognition: progress and good-practice benchmarking. Curr Opin Struct Biol. 2011;21(3): 404–11.
37. Yamada KD. Derivative-free neural network for optimizing the scoring functions associated with dynamic programming of pairwise-profile alignment. Algorithms Mol Biol. 2018;13:5.
38. Yamada KD, Tomii K, Katoh K. Application of the MAFFT sequence alignment program to large data-reexamination of the usefulness of chained guide trees. Bioinformatics. 2016;32(21):3246–51.
39. Henikoff S, Henikoff JG. Amino acid substitution matrices from protein blocks. Proc Natl Acad Sci U S A. 1992;89(22):10915–9.
40. Yamada K, Tomii K. Revisiting amino acid substitution matrices for identifying distantly related proteins. Bioinformatics. 2014;30(3):317–25.
41. Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, Baker L, Lai M, Bolton A. Mastering the game of go without human knowledge. Nature. 2017;550(7676):354.