A regulatory-sequence classifier with a neural network for genomic information processing

Koh Onimaru[1,2,✉], **Osamu Nshimura**[1,2], **and Shigehiro Kuraku**[1,2]

[1]Phyloinformatics Unit, RIKEN Center for Life Science Technologies (CLST), 2-2-3 Minatojima-minamimachi, Chuo-ku,Kobe, 650-0047, Japan
[2]Laboratory for Phyloinformatics unit, RIKEN Center for Biosystems Dynamics Research (BDR), 2-2-3 Minatojima-minamimachi, Chuo-ku, Kobe, Hyogo, 650-0047, Japan.

Genotype-phenotype mapping is one of the fundamental challenges in biology. The difficulties stem in part from the large amount of sequence information and the puzzling genomic code, particularly of non-protein-coding regions such as gene regulatory sequences. However, recently deep learning–based methods were shown to have the ability to predict gene regulatory sequences from genomes. Still, prediction accuracy needs improvement. Here, we report the design of convolution layers that efficiently process genomic sequence information and developed a software, DeepGMAP, to train and compare different deep learning–based models (https://github.com/koonimaru/DeepGMAP). First, we demonstrate that our convolution layers, termed forward- and reverse-sequence scan (FRSS) layers, enhance the power to predict gene regulatory sequences. Second, we assessed previous studies and identified problems associated with data structures that caused overfitting. Finally, we introduce several visualization methods that provide insights into the syntax of gene regulatory sequences.

gene regulatory sequence | convolutional neural network | DNase-seq | machine learning

Correspondence:
koh.onimaru@riken.jp

Introduction

In the last decade, advances in DNA sequencing technologies have dramatically increased the amount of genome sequence data derived from diverse species (1) as well as from individual humans (2).The next demanding challenge is the deeper understanding of how genome sequences encode organismal traits and how functional information can be extracted (3). Such sequence-based understanding would ultimately enable the prediction of phenotypes based on genome sequence information, i.e., genotype-phenotype mapping. The syntax of protein-coding genes is well understood, e.g., certain phenotypic consequences are predictable (such as nonsense mutations), yet the basic rules for non-coding sequences have not been completely established. Several projects including ENCODE (4–6), ROADMAP (7), and FANTOM (8) have accumulated epigenomic data to annotate the characteristics of non-coding sequences, but a comprehensive understanding of the association between epigenomic signatures and sequence information has yet to be attained.

Recently, this challenge has been addressed with deep learning–based methods (9–11). Deep learning is a subfield of machine-learning methods, and has been applied to a variety of problems such as image classification and speech recognition (12). Although much research has used deep-learning methods to automate routine but complex tasks that only humans were previously able to carry out, some cases are intended to surpass the human ability, such as playing the game "Go" (13). For the application of deep learning to genomics, convolutional neural networks (CNN) or recurrent neural networks (RNN) are trained with input genomic sequences that are labeled with epigenomic data to predict functional non-coding sequences. Once trained, these types of classifiers can infer the functional effect of mutations in non-coding genomic regions from individual genome sequences. However, even though these deep learning–based classifications have outperformed other methods, such as the support vector machine (14), the accuracy of prediction is still far from satisfactory. the prediction accuracy is still far from satisfactory. In addition, as it is generally known, it remains elusive as to what deep learning–based models actually "learn" and what kinds of "understanding" underlies their predictions (15, 16). In the present study, we first describe a CNN-based classifier that can outperform state-of-art models. Second, we identified problems associated with data structures that caused overfitting. Finally, we introduce methods to visualize trained models, potentially revealing the general syntax underlying regulatory sequences.

Results

To improve the accuracy of predicting regulatory sequences, we devised a deep learning–based method with three main features: a) integrating information from forward and reverse DNA sequences; b) simplifying the data structure; c) a new quality index to filter out low-quality data from a training dataset.

As training data, we downloaded several alignment files containing data from chromatin accessibility assays and chromatin immunoprecipitation-sequencing (ChIP-seq) experiments from the ENCODE project, and regions enriched with reads were determined as peaks by MACS2 peak caller (17). These data were used to mark genome sequences. We divided each of the mouse and human genome sequences into 1000-basepair (bp) windows and converted the four letters (i.e., A, C, G, T) into one-hot vectors (four dimensional zero-one vectors). To denote epigenomic marks, we assigned each window as 1 if it overlapped a signal positive region or 0 otherwise (signal negative) (see Methods for details). To reduce

DRAFT

the number of potential artifacts caused by window boundaries, we also added windows that were shifted by 500 bp toward the 3' side (Fig. 1a; we refer to this window structure as 1-kbp window/0.5-kbp stride). These data were used to train CNN models (see Supplementary Fig. 1 for the overall scheme).

To find an effective architecture of neural networks, we first compared the performance of published models such as CNN-based models (DeepSEA and Basset) and a CNN-RNN hybrid (DanQ; see Supplementary Table 1 for details of the models). As a quick benchmark, we trained them with a subset of mouse DNase-seq data (Supplementary Table 2) because DNase-seq data generally include diverse regulatory elements (18). To reduce computation time, we limited the number of training epochs to one, which means that models were trained with the full dataset only once. For evaluation, we used the entire chromosome 2 sequence, which was excluded from training datasets, and calculated the prediction accuracy of models using two scores, namely the area under the receiver-operation curve (AUROC) and the area under the precision-recall curve (AUPRC). Because AUROC is not a proper criterion if the majority of the data is negatively labeled, we considered AUPRC as a more important criterion (19). Consequently, the DeepSEA architecture performed better than Basset and DanQ (Table 1). The relatively poor performance of DanQ seemed inconsistent with a previous report (11), and this was probably attributable to an insufficient amount of training, i.e., the DanQ model requires 60 epochs to attain good performance (11).

Based on this result, we next explored whether better models could be attained by modifying the DeepSEA architecture. First, as more stratified convolutional networks are known to perform better (20), we inserted an additional convolution layer between the third convolution layer and the first fully connected layer of the DeepSEA model (with a reduced kernel number in some layers and smaller pooling patches). As expected, the four-layer convolutions achieved slightly better accuracy (conv4 in Table 1). We next focused on one of the fundamental characteristics of genome sequences—the forward and reverse strands. In previous studies, the information on reverse-strand sequences was ignored or used as independent data. However, for example, transcription factors with leucine-zipper or helix-loop-helix domains bind both strands through dimerization, and such binding thus results in palindromic binding motifs (21, 22). Inspired by this fact, we devised a set of layers that can integrate information from the both strands (Fig. 1b and Methods). We arranged the one-hot vectors that represent AGCT in a symmetric manner so that a 180-degree rotation of the one-hot vectors results in a reverse complementary sequence (Fig. 1c). Thus, reverse-sequence information can be processed by rotating the kernels 180 degrees. The hidden layers derived from the two strands are combined by element-wise addition after the second convolution. We termed this architecture FRSS (forward- and reverse-sequence scan) layers. The replacement of the first two convolutional layers in the DeepSEA model with FRSS (conv3-FRSS in Table 1) improved the accuracy of prediction more than the simple addition of a convolution layer (conv4). Moreover, a four-layer convolution model with FRSS achieved higher AUPRC scores than the other models (conv4-FRSS in Table 1, and see Supplementary Fig. 2 for a visual comparison). Raising the kernel numbers of each convolution layer also slightly improved the performance (conv4-FRSS+ in Table 1) but also increased computation time (the right most column in Table 1). We also found that the FRSS layers increased the learning efficiency of the DanQ model by a level comparable with conv4-FRSS (DanQ-FRSS in Table 1). These results were consistently reproduced by repeated training and testing (Supplementary Table 3), indicating that the benchmark is robust against random initialization. Together, these results showed that the FRSS layers enhance the predictive power of deep learning–based models.

Next, we examined the structure and quality of training data using the conv4-FRSS model. In certain previous studies, epigenetic signals were distributed into 200-bp windows of genome sequences, and 400-bp extra-sequences were added to the left and right sides of each window (10) (referred to as "DeepSEA-type data" in this study; Fig. 1a). For comparison, we trained conv4-FRSS with the DeepSEA-type data structure. Training with the DeepSEA-type data yielded a higher AUROC score but a lower AUPRC score (DeepSEA-type in Table 2). The high value of the AUROC is probably attributable to an increase in negative labels. In addition, we tested several window and stride sizes and found that models can be trained most efficiently with a 1-kbp window and 0.3-kbp stride data. As shown in Table 2, training with small-sized strides such as 0.1 and 0.2 kbp (and also DeepSEA-type data) yielded higher AUPRC scores when tested with the chromosome 1 sequence, which was included in the training data, than tested with the test dataset (chromosome 2, which was excluded from the training data), indicating overfitting to redundant sequences in the training dataset. This result also explains the unusually high AUPRC scores reported by Min et al. (2017) (23), which is inconsistent with the results of Zhou & Troyanskaya (2015) (10)). Because Min et al. trained CNN-based classifiers with 2-bp stride data and evaluated their model by resampling training data, the high AUPRCs in their report seemed to result from overfitting. We also examined peak callers. Although we utilized MACS2 to determine peak regions, previous studies used peak regions provided by the ENCODE project, which utilized Hotspot (24) as the peak caller. As shown in Table 2, training with Hotspot peaks yielded lower prediction accuracy (in Table 2). These results indicated that data-structure design is a critical factor for training models efficiently and that data augmentation using small strides, which was implemented in several studies (23, 25), is not an optimal strategy for training models.

To validate the aforementioned quick benchmark, we extended the list of training data. First, we newly trained conv4-FRSS with the same set of 125 human DNase-seq data used

DRAFT

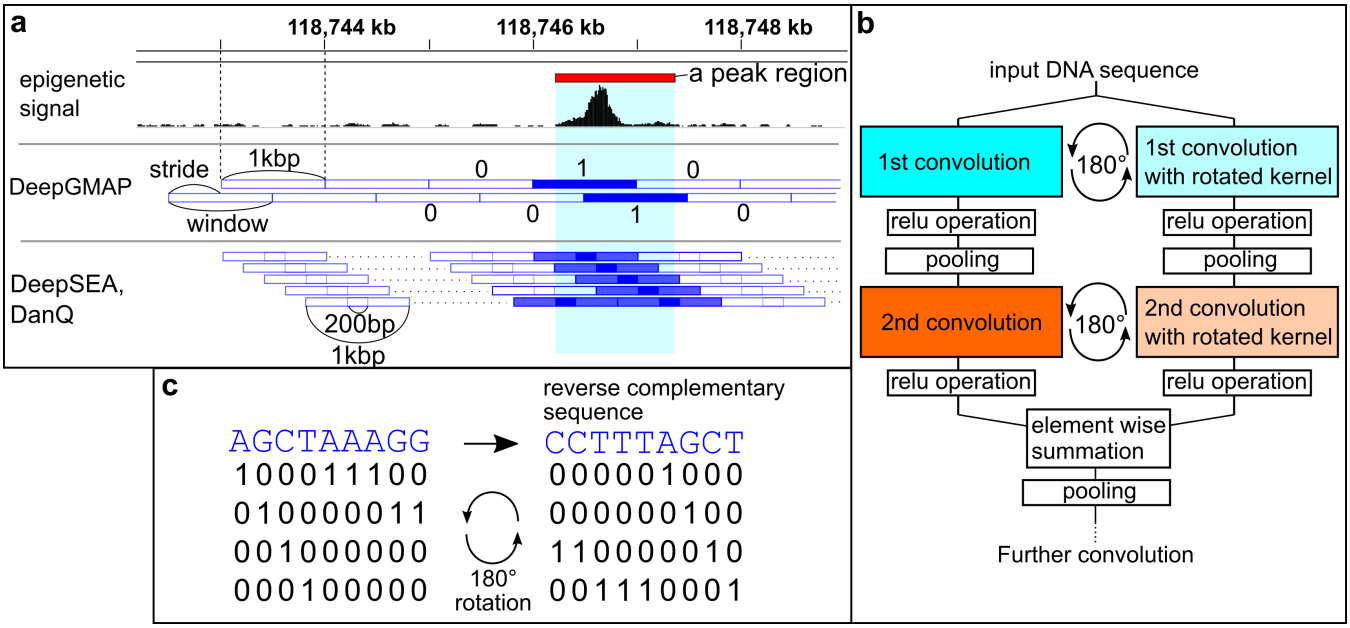

Fig. 1. The forward- and reverse-sequence scan (FRSS) layers and data structures. **a**, Subdivision of genome sequences into windows, and assignment of epigenomic signals. A peak region (red rectangle) is determined from epigenomic data and assigned as "1" to overlapping genomic regions (blue-filled rectangle), otherwise "0" (blue, empty rectangle). DeepGMAP, this study; DeepSEA, DanQ, previous studies. **b**,The architecture of the FRSS layers. An input sequence is scanned by the first convolution kernels, and the kernels are rotated in parallel. After implementing the relu operation, pooling, and the second convolution, the two parallel outputs are combined through summation. **c**, Illustration of the 180-degree rotation of kernels. In this example, a 9×4 kernel is trained to detect a sequence, AGCTAAAG (left). The geometric rotation of the kernel (right) results in the reverse complement of the original sequence.

Table 1. A performance comparison between models

	129_ES_E14		forebrain E11.5		forelimb E11.5		mean		time[b]
	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC	
DeepSEA	0.9234	0.6038	0.9141	0.6380	0.9315	0.5734	0.9230	0.6051	0.1773
Basset	0.9085	0.5796	0.9107	0.6280	0.9175	0.5476	0.9123	0.5851	0.0847
DanQ	0.9057	0.5785	0.9076	0.6214	0.9046	0.5164	0.9059	0.5721	0.4586
DanQ block[a]	0.9053	0.5699	0.9063	0.6168	0.9030	0.5031	0.9049	0.5633	0.4832
conv4	0.9300	0.6234	0.9146	0.6426	0.9351	0.5884	0.9266	0.6181	0.2548
conv3-FRSS	0.9364	0.6447	0.9171	0.6496	0.9421	0.6098	0.9319	0.6347	0.3636
conv4-FRSS	**0.9414**	**0.6571**	**0.9178**	**0.6520**	**0.9476**	**0.6280**	**0.9356**	**0.6457**	0.3729
conv4-FRSS+	**0.9418**	**0.6631**	**0.9183**	**0.6549**	**0.9475**	**0.6347**	**0.9359**	**0.6509**	0.5802
DanQ-FRSS	0.9365	0.6523	0.9162	0.6485	0.9453	0.6222	0.9327	0.6410	0.7905

[a]Tensorflow has several LSTM cells, and this model uses LSTMBlockCell, and the above uses LSTMCell. [b]the mean time (second) of 20 minibatch updates.

Table 2. A performance comparison between data structures

Data structure	chromosome 2 (true test)		chromosome 1 (overfitting test)	
	AUROC	AUPRC	AUROC	AUPRC
DeepSEA-type	0.9623	0.4841	0.9694	0.4741
1 kb window/0.5 kb stride[a]	0.9356	0.6457	0.9453	0.6293
1 kb window/0.4 kb stride	0.9352	0.6528	0.9457	0.6386
1 kb window/0.3 kb stride	**0.9398**	**0.6570**	**0.9515**	**0.6618**
1 kb window/0.2 kb stride	0.9397	0.6530	0.9581	0.6962
1 kb window/0.1 kb stride	0.9368	0.6396	0.9689	0.7757
Hotspot peaks	0.9287	0.6267	-	-

The mean values of the three mouse DNase-seq data are shown. The columns under chromosome 2 are the results of true tests (the higher is better). The columns under chromosome 1 are overfitting tests (if the scores are higher than those of true tests, that is the sign of overfitting). [a]The same data with the mean values of conv4-FRSS in Table 1.

DRAFT

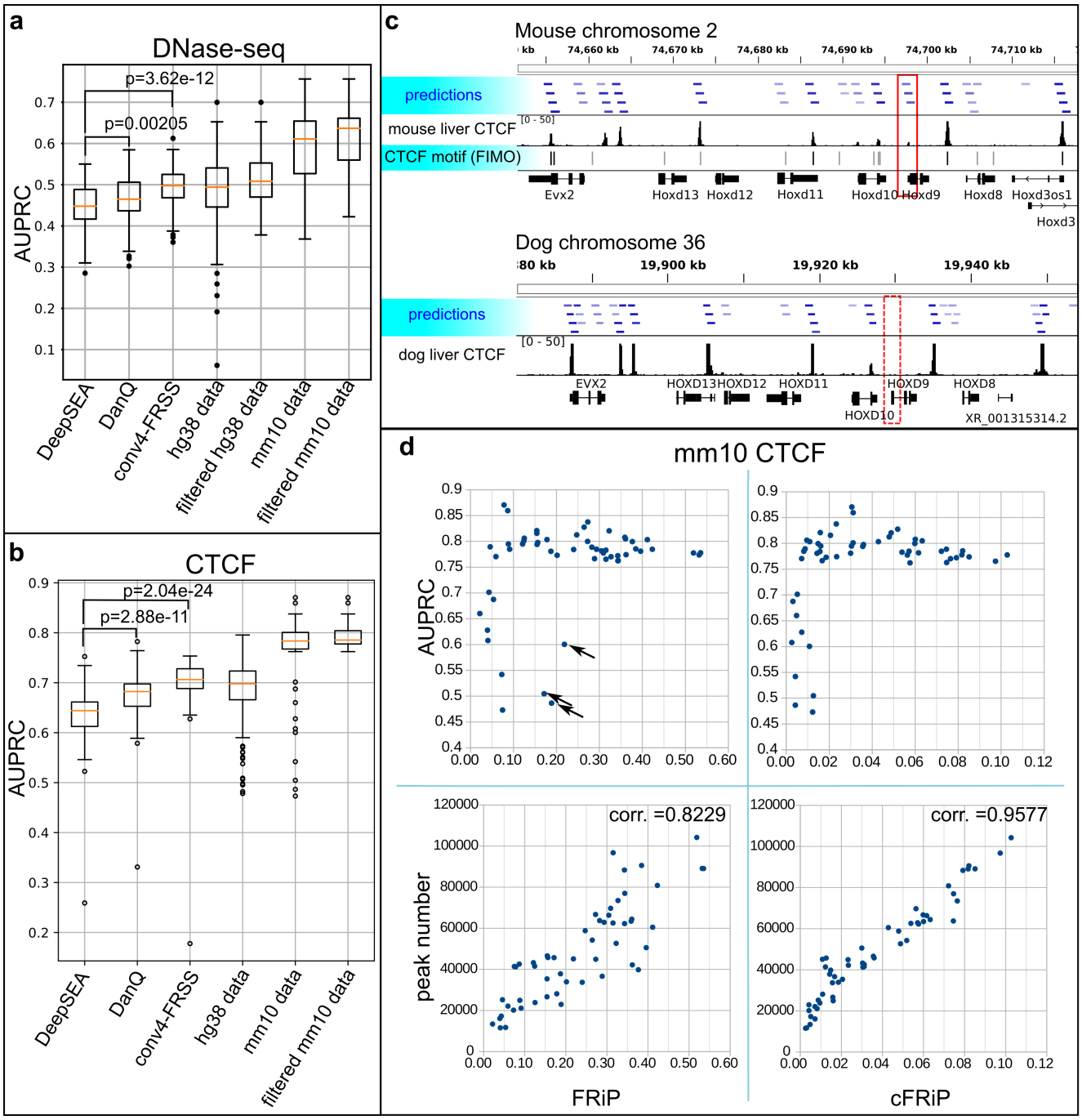

Fig. 2. Performance analyses of models with extended data sets. **a, b**, Comparison of AUPRCs of DNase-seq data (**a**) and CTCF data (**b**). DeepSEA and DanQ, AUPRCs obtained from their original reports; conv4-FRSS, predictions trained with the same peak files, but a different data structure (1-kbp window/0.3-kbp stride); hg38 and mm10 data, 361 and 93 DNase-seq datasets generated by the MACS2 peak caller; filtered hg38 and mm10 data, DNase-seq data with high cFRiPs. p, two-sided Wilcoxon–Mann–Whitney test.Boxes, the lower to upper quantile values of the data; orange lines, the median; whiskers, the range of the data (Q1 – IQR × 1.5 and Q3 – IQR × 1.5 for the lower and upper bounds, respectively); flier points, those past the end of the whiskers. **c**, An example of CTCF binding site predictions on the HoxD cluster of mice and dogs. Regions with prediction values >= 0.2 are visualized. Blue bars, CTCF binding sites predicted by conv4FRSS (the bluer is more probable); mouse/dog liver CTCF, alignment data from ENCFF627DYN and ERR022304 (26), respectively; respectively; CTCF motif (FIMO), sites that were detected by FIMO (black boxes, p <= 1e-4; gray boxes, 1e-4 < p <= 1e-3); red rectangles, a CTCF peak that is present in the mouse genome (solid rectangle) but not the dog genome (dashed rectangle). **d**, Scatter plots for FRiP and corrected FRiP with AUPRCs (top panels) and peak numbers (bottom panels). corr., Spearman correlation.

by Zhou & Troyanskaya (2015) (10) (see Supplementary Table 2 for the lists of data). We found that conv4-FRSS significantly outperformed DeepSEA and DanQ (Fig. 2a and Supplementary Table 4; the scores of DeepSEA and DanQ were obtained from their publications). We also trained conv4-FRSS with 365 and 93 of human and mouse DNase-seq peaks, respectively, generated by MACS2 (see Supplementary Table 2 for the lists of data). As shown in Fig. 2a and Supplementary Table 4, we noticed that training with mouse data yielded higher performance than with human data. The cause may be attributable to a difference in genome size, genetic variation (should be lower for inbred mice), or species-specific trends such as gene density and GC content. We also focused on CTCF ChIP-seq data because (a) CTCF has a general function in genome organization, (b) ENCODE releases plenty of CTCF data for both mice and humans, and (c) as evident for the AUPRC scores of DeepSEA (Supplementary Fig. 3; reanalyzed data from Zhou & Troyanskaya, 2015 (10)), the prediction accuracy is highly dependent on the target of ChIP-seq, and that of CTCF is the highest among transcription factors. As a result, the conv4-FRSS model was efficiently trained with both human and mouse CTCF data, and this model yielded a median AUPRC score of >0.70 (see Fig. 2b and 2c for a visual comparison between predictions and original signals). A comparison between the predictions of the model and CTCF motif regions detected by FIMO (27) [CTCF motif (FIMO) in Fig. 2c] revealed that the model can predict CTCF binding regions that do not contain the canonical CTCF motif. In addition, we also used the class saliency extraction method (28) to identify informative sequences at the single-nucleotide level (Supplementary Fig. 4). This method is equivalent to the in silico saturation mutagenesis approach used previously (9, 10) but is more computationally efficient because it does not require mutation-by-mutation evaluation (see Methods for details). Furthermore, using the dog genome (CanFam3.1), we performed a cross-species prediction of CTCF binding regions with the model trained solely with mouse data. As shown in the bottom panel of Fig. 2c, the predictions of the model matched with the real CTCF signals of dog liver (26) and detected species-specific differences (red rectangles in Fig. 2c), indicating the generality of the model. Based on all these data, we concluded that conv4-FRSS can be applied to diverse data.

In the above training with extended dataset, we suspected that several classes with low AUPRCs were attributable to poor data quality. To address this possibility, we calculated a known quality index, FRiP (fraction of reads in peaks (29)) of mouse CTCF data and found that classes with low AUPRCs tended to have low FRiPs (Fig. 2d), with some exceptions (arrows in Fig. 2d). Because FRiP is calculated based on reads in peaks per total reads, values may be inflated when the read number of the source data is too low. To correct such inflation, we multiplied FRiP by the fraction of genomic regions covered by at least one read in a genome. As shown in Fig. 2d, the corrected FRiPs (cFRiPs) could distinguish high and low AUPRC data more clearly. In addition, cFRiP yielded a stronger correlation with the total peak numbers of data than the uncorrected FRiP (Fig. 2e and Supplementary Table 5). Although the other datasets showed weaker relations between FRiP and AUPRC, data with optimal cutoff values of cFRiP increased the average AUPRC score except for the human CTCF data (filtered hg38 and mm10 data in Fig. 2a and 2b, and Supplementary Fig. 5). These results suggested two points. First, the correction method reasonably represents the true quality of the data. Second, the linear correlation between peak numbers and cFRiPs implies that data quality is not saturated enough to detect all true peaks (and hence, the possibility remains that the model learns sample quality rather than cell-specific patterns). Therefore, whereas the ENCODE project seems to consider $\geq$1% FRiP as good data (29), our results suggest that higher stringency during quality control is required for the precise annotation of genomes.

To obtain insights into how the models predict regulatory sequences and their syntax, we visualized the model (conv4-FRSS) trained with the subset of mouse DNase-seq data. First, as has been done in previous studies (9, 11),we analyzed individual kernels of the first layer, which directly interact with DNA sequences and thus represent DNA motifs that are important for classifying regulatory sequences. We converted the weight variables to probability matrices by using a softmax-like operation and found that established models have indeed learned many known transcription factor binding motifs, such as those bound by CTCF, Sox9, OCT4 and KLF4 (Fig. 3a, and Supplementary Fig. 6). Because gene regulation often requires the participation of several transcription factors (30), we visualized linkages between kernels and layers (Fig. 3b is linked to the 129 ES E14 class of the three DNase-seq data as an example, and Supplementary Fig. 7 for larger images of linkages to three classes). A comprehensive visualization of linkages is impossible owing to the large numbers of kernels and variables. Therefore, we calculated the strength of linkages by summing weight variables for each kernel, and top 10% of linkages to each class is shown in Fig. 3b and Supplementary Fig. 7. This partial visualization revealed that the information from the first layer ("hidden1") was gathered at relatively few nodes of the last convolution layer ("hidden4"). We also found that most of the strongly contributing kernels were shared between three classes. In addition, even if a kernel was found to be shared between two classes, the linkages to the last layers differed (kernels with a light blue rectangle and red colored linkages in Supplementary Fig. 7), suggesting that different combinations of kernels were used for the class-specific predictions. Another interesting point is the temporal development of kernels and connections. As shown in Supplementary Fig. 8, the model seems to learn GC-rich sequences (probably recognizing CpG islands or the CTCF binding motif) in early training stages, and kernels are connected by relatively few nodes of the second layer ("hidden2"). In later stages, the model increases the variation and connection of kernels. Taken together, these visualizations suggested that the model seems to discern regulatory sequences via analysis of a combination of kernels.

DRAFT

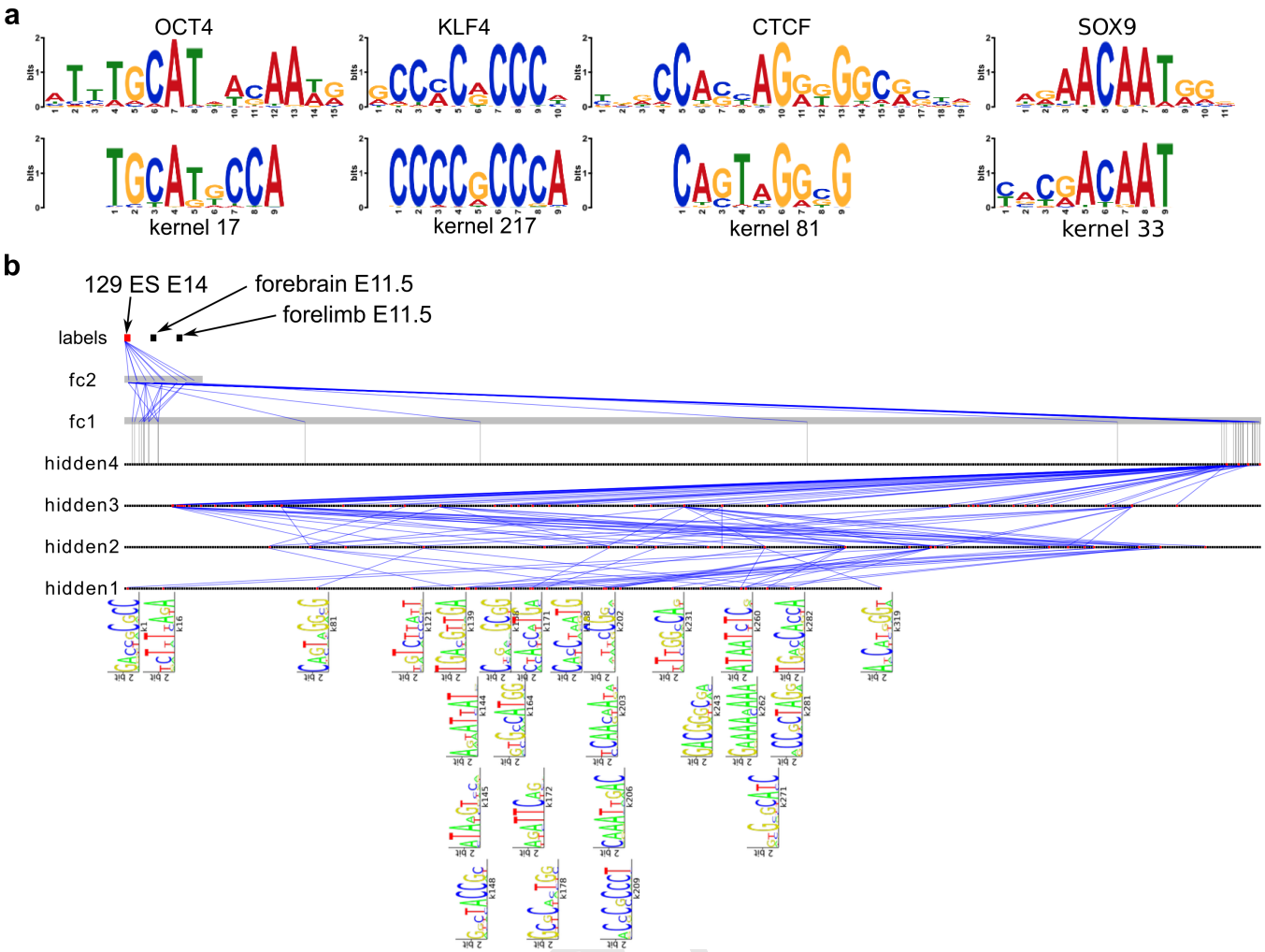

Fig. 3. Visualization of trained kernels and their connections. **a**, Examples of known motifs (top) that match with kernels. **b**, Connections of kernels and layers of conv4 with FRSS. As an example, connections that are linked to the 129 ES E14 DNase-seq class are shown. Dots of hidden1-4, the outputs of convolution operations of kernels; fc1 and fc2, the fully connected layers (because there were too many neurons, they are represented as gray bars); labels, the last output layer; blue lines, connections between kernels and layers. Larger images are available in Supplementary Fig. 6.

Next, we applied the activation maximization method25,29,30, in which we look for DNA sequences that maximize neuron activities in the final layer of a trained model by training the sequence itself (see Methods for formal mathematical expressions). First, we trained a sequence to activate the 129 ES E14–specific neuron in the last output layer. As with image-recognition studies (28, 31, 32), this method generated randomly repeated sequences that probably capture the 129 ES E14 class-specific traits (left panel in Fig. 4). Using the motif comparison software Tomtom (33), we found that the generated sequence contained several OCT4/SOX2 and KLF4 motifs, which are important for the pluripotency of embryonic stem cells (34). For comparison, we also trained a sequence to activate all of the three classes of the neurons, resulting in a CTCF motif–rich sequence (right panel in Fig. 4). This result is consistent with the general role of CTCF as an enhancer looping factor (35). In addition, the generated sequence also contained small motifs next to the well-known 20-bp motif (dashed boxes in the right panel of Fig. 4), which resembled the other part of the alternative long CTCF motif (33/34 bp in length, including an additional GGNANTGCA or TGCANTNCC sequence) (26). Thus, the detection of motifs that are longer than the kernel size constitutes one of the advantages of this activation maximization method. Taken together, these visualization methods would be useful for understanding how deep-learning models predict enhancer sequences.

Discussion

WOur results demonstrate that a new convolution architecture, FRSS, efficiently discerns patterns of regulatory DNA sequences. Furthermore, the analyses of data quality and structure revealed that training efficiencies highly depend on the means by which the source data are filtered and processed. Although the visualization methods are still in development, these methods will provide researchers with clues for understanding the syntax of gene regulatory sequences.

We designed FRSS to process the information for forward and reverse sequences simultaneously, and we demonstrated the efficacy of FRSS for this purpose. Contrary to other studies, this is the first to devise a new CNN architecture spe-

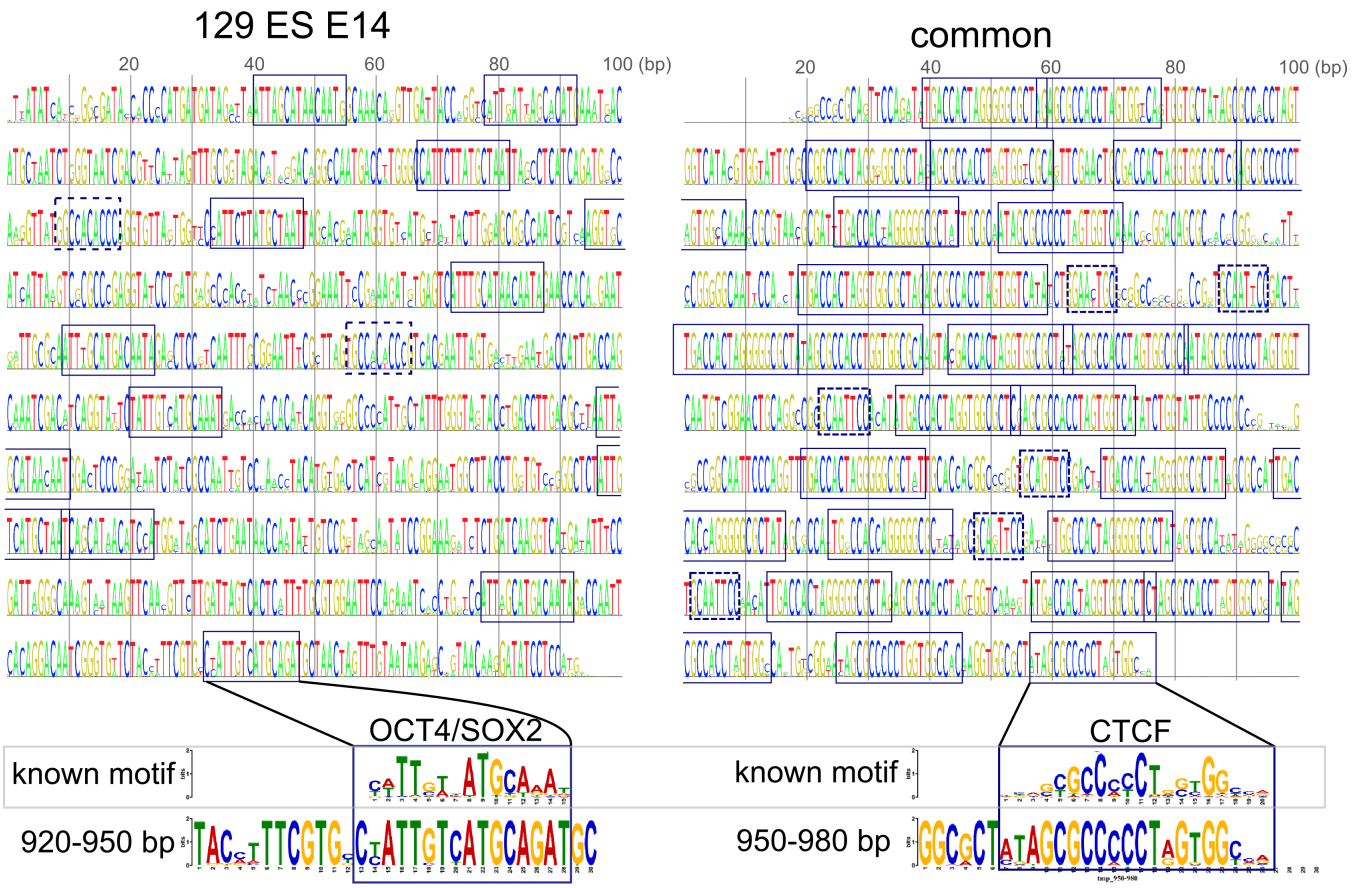

Fig. 4. Results of the activation maximization method. Sequences trained to activate the 129 ES E14–specific neuron in the last layer (left panel) and all neurons in the last layer (right panel), respectively. Boxes in the left panel, OCT4/SOX2 binding motifs; dashed boxes in the left panel, KLF4 binding motifs; boxes in the right panel, CTCF binding motifs; dashed boxes, motifs similar to the second part of the alternative longer CTCF motif.

cialized for processing DNA sequences. However, there are many ways for designing layers for this purpose. For example, instead of pair-wise summation in the last layer of FRSS, one can concatenate the output tensors similar to bidirectional RNN (36). Although we choose the current architecture because its computational cost is smaller than that of other tested architectures, better architectures may be designed that retain low computational cost yet offer greater prediction power. In conclusion, our study constitutes a new paradigm for developing deep learning–based sequence classifiers.

Methods

Dataset and processing. The sequences for the human (hg38) and mouse (mm10) genomes were downloaded from UCSC (http://hgdownload.soe.ucsc.edu/goldenPath/). Mitochondrial DNA sequences were excluded from analyses. The genomic sequences were divided into 1-kbp windows with a variety of strides as described in Results. The filtered alignment files for epigenomic data were downloaded from the ENCODE website (https://www.encodeproject.org/; see Supplementary Table 2 for details). A peak caller, MACS2 version 2.1.1.20160309, was used to determine signal peak regions with the following options: "callpeak –call-summits -t <target bam file> -c <control bam file>-f <BAM or BAMPE> -g <hs or mm> -q 0.01" for ChIP-seq, and "callpeak –call-summits -q 0.01 –nomodel –shift -100 –extsize 200 -t <target bam file> -f <BAM or BAMPE> -g <hs or mm>" for DNase-seq. Using the outputs of MACS2, bedtools (37), and our codes, genomic regions were designated as positive if a window was found to overlap with the summit of a peak. Data for which there was less than 10,000 peaks were excluded from training data. For comparative analyses in Fig. 2a and 2b, peak files were downloaded from http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/. For cross-species comparison, CanFam 3.1 was downloaded from NCBI (https://www.ncbi.nlm.nih.gov/genome/?term=dog), and the short reads for CTCF ChIP-seq data (ERR022304) were downloaded from EMBL-EBI ArrayExpress (https://www.ebi.ac.uk/arrayexpress/experiments/E-MTAB-437/).

The DNA symbols, A, G, C, T and N were converted into one-hot vectors: (1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0), (0, 0, 0, 1), and (0, 0, 0, 0), respectively. Therefore, one training sample became a 1000×4 tensor. With a mini-batch number of 100 for the stochastic gradient descent and channel dimension 1 (a dimension for colors if the input was image data), the final shape of the input tensor was 100×1000×4×1, with a label tensor of size mini-batch number×class number [i.e.,

100 label vectors, and each label looks like (0, 0, 0, 1, 0, 1, ..., 0); see Supplementary Fig. 1 for visual illustration].

Design of models and implementation. Tensorflow r1.8 for python (https://www.tensorflow.org/) with CUDA Driver Version 9.0 and cuDNN v7.0.5 was used as the main machine-learning library for model implementation. The Tensorflow library was compiled from the source codes with the bazel compiler (https://bazel.build/; with options: "-c opt --copt=-mavx --copt=-mavx2 --copt=-mfma --copt=-mfpmath=both --copt=-msse4.2 --config=cuda -k //tensorflow/tools/pip_package:build_pip_package"). For convolutions, we used a module, tensorflow.nn.conv2d, with options: "strides=[1, 1, 1, 1], padding=VALID".For the Basset model, because batch normalization did not yield any positive effects, we removed this operation. For RNN layers, we used the long short-term memory cells (38) (tensorflow.nn.rnn_cell.LSTMCell or tensorflow.contrib.rnn.LSTMBlockCell without the peephole option), and tensorflow.nn.bidirectional_dynamic_rnn for output calculations. The operations and hyperparameters of models are listed in Supplementary Table 1.

For the FRSS layers, convolution kernels were rotated 180 degrees to extract reverse complementary information in the first and second layers in parallel with normal convolutions as follows. Let W^{ℓ} a weight tensor of size M×N×H×K (M, kernel height; N, kernel width; H, input channel number; K, output channel number) in the ℓth convolution layer, defined as $w^{\ell}_{m,n,h,k}$ ($m = 0, ..., M-1$; $n = 0, ..., N-1$; $h = 0, ..., H-1$; $k = 0, ..., K-1$). The rotated tensor W^{ℓ}_{rc} can be written as $[W^{\ell}_{rc}]_{m,n,h,k} = w^{\ell}_{(M-1-m),(N-1-n),h,k}$. Then, the first part of the FRSS layers is

$$X^1 = pool(relu(conv(W^0, X^0))),$$
$$X^1_{rc} = pool(relu(conv(W^0_{rc}, X^0))),$$

where X^{ℓ} is a input/output tensor shape of B×S×N×H (B, mini batch number; S, sequence length; N, input width; H, input channel number) in the ℓth convolution layer, X^{ℓ}_{rc} is the output of convolution with the rotated tensor, *conv* is the convolution operation, *relu* is the rectified linear unit, and *pool* is the max pooling operation. The second part of the FRSS layers is

$$X^2 = pool(relu(conv(W^1, X^1)) + relu(conv(W^1_{rc}, X^1))).$$

Note that the variables of W^{ℓ}_{rc} are not independent training targets, but rather are the copies of those of W^{ℓ} .

Training and testing models. Stochastic gradient–based optimization was used for training models. For human data, chromosomes 8 and 9 were excluded from training data. For mouse data, chromosome 2 was excluded. The excluded sequences were used to test trained models. hg19 genome was used for comparisons with previously reported AUPRCs (Fig. 2a and b). A training dataset was subdivided into mini-batches containing 100 sequences. In the quick benchmark, because the fraction of positively labeled genomic windows was too small, negative samples were randomly downsampled to 25%. To accelerate the learning rate and compensate for the limited data size, each mini-batch was further divided into two sub-mini-batches, and updates were made alternately twice per sub-mini-batch (i.e., four updates per mini-batch). The loss function we used for training our models and the DeepSEA model is

$$\frac{1}{n}\sum_{k=0}^{n-1} y_k \log(\hat{y}_k) + (1-y_k)\log(1-\hat{y}_k) + \lambda_1 \|W\|_2^2 + \lambda_2 \|H\|_1,$$

where y_k is labels, $\hat{y}_k$ is predictions after the sigmoid operation, $\|W\|_2^2$ is the L2 regularization of all variables, $\|H\|_1$ is the L1 regularization of predictions before the sigmoid operation, λ_1 is 5×10^{-7}, and λ_2 is 1×10^{-8}. For the remaining models, the loss function without the regularization terms was used. The optimization algorithms RMSprop (39) and Adam (40) were used to train the DanQ model and the others, respectively. To monitor training accuracy, models were tested by every mini-batch before updating variables and then evaluated based on the F1 score ($\frac{2 \cdot precision \cdot recall}{precision + recall}$) to assess temporal accuracy. When a mean of F1 scores of the last three tests exceeded a certain threshold (0.75), models were tested by three mini-batch sets that had been randomly excluded from a training dataset and saved when a higher F1 score was attained relative to the previous test.

NVIDIA TITAN X (Pascal; 3584 CUDA Cores; total amount of global memory, 12190 Mbytes; GPU Max Clock rate, 1531 MHz) was used for GPU-computing. Intel® Xeon® Processor E5-2640 v4 (2.40 GHz) was used for CPU-computing (see Supplementary Notes for the full specification of our machine). Total training time was dependent on the number of classes, the amount of training data, and model architecture. For example, conv4-FRSS took 3 hours to train with the subset of mm10 DNase-seq data (three classes) and 11 hours with hg38 DNase-seq (365 classes). The bottleneck was, in part, to send tensor data from the GPU to CPU for evaluation of the temporal accuracy of the training models. In previous studies, training time was reported as 85 hours in the Basset paper with NVIDIA Tesla K20m and 164 classes, and as approximately 15 days in the DanQ paper with NVIDIA Titan Z and 919 classes. However, a fair comparison of training time with previous studies is difficult owing to differences in the machines, languages, and datasets.

To test trained models, the chromosome 2 sequence for mice and chromosome 8 and 9 sequences for humans were divided into the same window size and stride with those of the training data. AUROC and AUPRC were calculated with functions in the scikit-learn library version 0.19.1 (http://scikit-learn.org/stable/). We compared the accuracy between a model trained with the full training data and one saved during training, and adopted the one with higher AUPRCs. AUPRCs of DeepSEA and DanQ in Fig. 2a and b were obtained from Supplementary Table 3 of the

DRAFT

DeepSEA paper (https://media.nature.com/original/nature-assets/nmeth/journal/v12/n10/extref/nmeth.3547-S3.xlsx) and auc.txt of the DanQ repository (https://github.com/uci-cbcl/DanQ) and are listed in our Supplementary Table 4. Predictions presented in Fig. 2c and Supplementary Fig. 2 were visualized using the integrative genomics viewer (41).

The class saliency extraction method. To evaluate informative sequences at the single-nucleotide level (Supplementary Fig. 4), the class saliency extraction method was performed as previously described (28) with slight modifications. Let $\hat{y}_i\left(X^0\right)$ be the score of the class *i*, computed by the conv4-FRSS model for an arbitrary DNA sequence X^0 of size 1000×4×1. The derivative of $\hat{y}_i\left(X^0\right)$ with respect to X^0 is

$$\omega = \left.\frac{\partial \hat{y}_i\left(X^0\right)}{\partial X^0}\right|_{X_0^0},$$

where X_0^0 is a given DNA sequence. The derivative ω is a tensor of size 1000×4×1, and each element corresponds to each element of X^0. The values shown in Supplementary Fig. 4a (orange) are $\underset{n}{max}\left|\omega_{s,n,h}\right|$ (s=0, . . . , 999; *n*=0, ..., 3; *h*=0). These values represent the magnitude of the effect on the class score when nucleotide substitutions occur. Therefore, these values may be termed as influential values.

FRiP calculation and correction. To calculate FRiPs, initially, mitochondrial and black-listed regions (https://sites.google.com/site/anshulkundaje/projects/blacklists) were removed from peak files. A module, "countReadsPerBin.CountReadsPerBin" in deeptools (42) was used to count reads in peaks, and these read counts were then divided by total reads. As a correction for read number differences, FRiP was multiplied by the fraction of genome regions covered by at least one read.

Visualizing the inside of a trained model. Variables in each kernel were converted to a probability matrix by a softmax-like operation. Tomtom (33) version 4.12.0 and meme motif database version 12.15 (http://meme-suite.org/meme-software/Databases/motifs/motif_databases.12.15.tgz) were used to identify closely related known transcription factor binding motifs. For visualizing kernels in the first layer of FRSS, the probability matrices were scaled by information content. Motif logos except Tomtom outputs were generated by our customized codes using the cairocffi library (https://cairocffi.readthedocs.io/en/latest/).

To visualize the connections of kernels and layers, we examined how values flow in the network, as follows. The first input sequence is a tensor of size B×1000×4×1 or 1000×4×1 (B is the minibatch number and can be ignored here). By the first convolution, in which each of 320 kernels has a 9×4 matrix, the number of output sequences is 320, and each sequence length becomes 982×1 (982×1×320). After 2×1 max pooling, the sequence length is reduced to 496×1×320. Therefore, "hidden1" in Fig. 3b and Supplementary Fig. 6 is composed of 320 dots, and each dot contains 496×1 values. Because these 320 dots are the outputs of 320 kernels, each dot has a corresponding kernel (the bottom part of Fig. 3B, but strongly connected kernels to “hidden2" are shown). From "hidden1" to "hidden2", he number of kernels is 480, and each kernel has 320 matrices of size 9×1. Each of the 320 matrices in a kernel operates a convolution with one of the 320 outputs from the first convolution. Then, each kernel sums the 320 convolution results, which generates 480 outputs (the dots of "hidden2" in Fig. 3b) with a sequence of length 488 in each (i.e., a tensor of size 488×1×480). Namely, if a matrix of a kernel has large positive values, a corresponding output from the first convolution (a dot of "hidden1") strongly contributes to one of 480 outputs (a dot of "hidden2"). Therefore, the sum of 9 values in a matrix of a kernel approximates the strength of a connection between two dots of "hidden1" and "hidden2". The same algorithm was applied to the other convolution layers. The connections between the fully connected layers are simpler, as each variable represents one connection between layers. The figures present only the top 10% of the strong connections. The networks were written by our customized codes using the cairocffi library.

The activation maximization method. Let $\hat{y}_i(X^0)$ be the score of the class *i* $i = 0, 1, 2$, computed by the conv4-FRSS model for a sequence X^0. To find a sequence that is specific to the class *i*, we sought to optimize the following problem:

$$\underset{X^0}{argmax}\frac{\hat{y}_i(X^0)}{\sum_{k\neq i}\hat{y}_k(X^0)} - \lambda\|X^0\|_2^2,$$

where λ is 5.0×10^{-3}. For the common sequence (right panel in Fig. 4), the problem was:

$$\underset{X^0}{argmax}\prod\hat{y}_i(X^0) - \lambda\|X^0\|_2^2.$$

These formulae differ slightly from those reported previously (28, 32) because our model used the sigmoid function for the last output, whereas the others used softmax. The optimizer Adam was used to optimize the target sequence. The initialization of variables of $\boldsymbol{X^0}$ was set to have a mean of 0.02 and stddev of 0.02, and this was a critical factor for this optimization problem. Training with different initialization conditions sometimes did not converge, suggesting that the optimization landscape is rugged and there is a possibility that the results shown in Fig. 4 are not the optimal solution. The generated sequences were visualized with our customized codes using the cairocffi library. Tomtom (33) was used to find motifs in the generated sequences.

Code availability

All codes used in this paper is available at https://github.com/koonimaru/DeepGMAP.

Data availability The IDs for data downloaded from the ENCODE website are listed in Supplementary Table 2. The data generated and/or analyzed in the current study are available in the figshare repository, https://doi.org/10.6084/m9.figshare.6728348.

ACKNOWLEDGEMENTS

We thank Dr. Yuichiro Hara for critical comments and Dr. Mitsutaka Kadota for discussions concerning unpublished data. This work was supported in part by JSPS KAKENHI grant number 17K15132, a Special Postdoctoral Researcher Program of

DRAFT

RIKEN, and a research grant from MEXT to the RIKEN Center for Life Science Technologies and RIKEN Center for Biosystems Dynamics Research.

Bibliography

1. Jennifer R.S. Meadows and Kerstin Lindblad-Toh. Dissecting evolution and disease using comparative vertebrate genomics, 2017. ISSN 14710064.
2. 1000 Genomes Project Consortium, Goncalo R Abecasis, Adam Auton, Lisa D Brooks, Mark A DePristo, Richard M Durbin, Robert E Handsaker, Hyun Min Kang, Gabor T Marth, and Gil A McVean. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491(7422):56–65, 2012. ISSN 1476-4687. doi: 10.1038/nature11632.
3. Claudius F. Kratochwil and Axel Meyer. Closing the genotype-phenotype gap: Emerging technologies for evolutionary genetics in ecological model vertebrate systems. *BioEssays*, 37(2):213–226, 2015. ISSN 15211878. doi: 10.1002/bies.201400142.
4. Feng Yue, Yong Cheng, Alessandra Breschi, Jeff Vierstra, Weisheng Wu, Tyrone Ryba, Richard Sandstrom, Zhihai Ma, Carrie Davis, Benjamin D. Pope, Yin Shen, Dmitri D. Pervouchine, Sarah Djebali, Robert E. Thurman, Rajinder Kaul, Eric Rynes, Anthony Kirilusha, Georgi K. Marinov, Brian A. Williams, Diane Trout, Henry Amrhein, Katherine Fisher-Aylor, Igor Antoshechkin, Gilberto DeSalvo, Lei Hoon See, Meagan Fastuca, Jorg Drenkow, Chris Zaleski, Alex Dobin, Pablo Prieto, Julien Lagarde, Giovanni Bussotti, Andrea Tanzer, Olgert Denas, Kanwei Li, M. A. Bender, Miaohua Zhang, Rachel Byron, Mark T. Groudine, David McCleary, Long Pham, Zhen Ye, Samantha Kuan, Lee Edsall, Yi Chieh Wu, Matthew D. Rasmussen, Mukul S. Bansal, Manolis Kellis, Cheryl A. Keller, Christapher S. Morrissey, Tejaswini Mishra, Deepti Jain, Nergiz Dogan, Robert S. Harris, Philip Cayting, Trupti Kawli, Alan P. Boyle, Ghia Euskirchen, Anshul Kundaje, Shin Lin, Yiing Lin, Camden Jansen, Venkat S. Malladi, Melissa S. Cline, Drew T. Erickson, Vanessa M. Kirkup, Katrina Learned, Cricket A. Sloan, Kate R. Rosenbloom, Beatriz Lacerda De Sousa, Kathryn Beal, Miguel Pignatelli, Paul Flicek, Jin Lian, Tamer Kahveci, Dongwon Lee, W. James Kent, Miguel Ramalho Santos, Javier Herrero, Cedric Notredame, Audra Johnson, Shinny Vong, Kristen Lee, Daniel Bates, Fidencio Neri, Morgan Diegel, Theresa Canfield, Peter J. Sabo, Matthew S. Wilken, Thomas A. Reh, Erika Giste, Anthony Shafer, Tanya Kutyavin, Eric Haugen, Douglas Dunn, Alex P. Reynolds, Shane Neph, Richard Humbert, R. Scott Hansen, Marella De Bruijn, Licia Selleri, Alexander Rudensky, Steven Josefowicz, Robert Samstein, Evan E. Eichler, Stuart H. Orkin, Dana Levasseur, Thalia Papayannopoulou, Kai Hsin Chang, Arthur Skoultchi, Srikanta Gosh, Christine Disteche, Piper Treuting, Yanli Wang, Mitchell J. Weiss, Gerd A. Blobel, Xiaoyi Cao, Sheng Zhong, Ting Wang, Peter J. Good, Rebecca F. Lowdon, Leslie B. Adams, Xiao Qiao Zhou, Michael J. Pazin, Elise A. Feingold, Barbara Wold, James Taylor, Ali Mortazavi, Sherman M. Weissman, John A. Stamatoyannopoulos, Michael P. Snyder, Roderic Guigo, Thomas R. Gingeras, David M. Gilbert, Ross C. Hardison, Michael A. Beer, and Bing Ren. A comparative encyclopedia of DNA elements in the mouse genome. *Nature*, 515(7527):355–364, 2014. ISSN 14764687. doi: 10.1038/nature13992.
5. Bradley E Bernstein, Ewan Birney, Ian Dunham, Eric D Green, Chris Gunter, and Michael Snyder. An integrated encyclopedia of DNA elements in the human genome. *Nature*, 489: 57–74, 2012. ISSN 1476-4687. doi: 10.1038/nature11247.
6. Cricket A. Sloan, Esther T. Chan, Jean M. Davidson, Venkat S. Malladi, J. Seth Strattan, Benjamin C. Hitz, Idan Gabdank, Aditi K. Narayanan, Marcus Ho, Brian T. Lee, Laurence D. Rowe, Timothy R. Dreszer, Greg Roe, Nikhil R. Podduturi, Forrest Tanaka, Eurie L. Hong, and J. Michael Cherry. ENCODE data at the ENCODE portal. *Nucleic Acids Research*, 44 (D1):D726–D732, 2016. ISSN 13624962. doi: 10.1093/nar/gkv1160.
7. Bradley E. Bernstein, John A. Stamatoyannopoulos, Joseph F. Costello, Bing Ren, Aleksandar Milosavljevic, Alexander Meissner, Manolis Kellis, Marco A. Marra, Arthur L. Beaudet, Joseph R. Ecker, Peggy J. Farnham, Martin Hirst, Eric S. Lander, Tarjei S. Mikkelsen, and James A. Thomson. The NIH roadmap epigenomics mapping consortium, 2010. ISSN 10870156.
8. Robin Andersson, Claudia Gebhard, Irene Miguel-Escalada, Ilka Hoof, Jette Bornholdt, Mette Boyd, Yun Chen, Xiaobei Zhao, Christian Schmidl, Takahiro Suzuki, Evgenia Ntini, Erik Arner, Eivind Valen, Kang Li, Lucia Schwarzfischer, Dagmar Glatz, Johanna Raithel, Berit Lilje, Nicolas Rapin, Frederik Otzen Bagger, Mette Jørgensen, Peter Refsing Andersen, Nicolas Bertin, Owen Rackham, A. Maxwell Burroughs, J. Kenneth Baillie, Yuri Ishizu, Yuri Shimizu, Erina Furuhata, Shiori Maeda, Yutaka Negishi, Christopher J. Mungall, Terrence F. Meehan, Timo Lassmann, Masayoshi Itoh, Hideya Kawaji, Naoto Kondo, Jun Kawai, Andreas Lennartsson, Carsten O. Daub, Peter Heutink, David A. Hume, Torben Heick Jensen, Harukazu Suzuki, Yoshihide Hayashizaki, Ferenc Müller, Alistair R.R. Forrest, Piero Carninci, Michael Rehli, and Albin Sandelin. An atlas of active enhancers across human cell types and tissues. *Nature*, 507(7493):455–461, 2014. ISSN 14764687. doi: 10.1038/nature12787.
9. David R. Kelley, Jasper Snoek, and John L. Rinn. Basset: Learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome Research*, 26(7): 990–999, 2016. ISSN 15495469. doi: 10.1101/gr.200535.115.
10. Jian Zhou and Olga G Troyanskaya. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature Methods*, 12(10):931–934, 2015. ISSN 1548-7091. doi: 10.1038/nmeth.3547.
11. Daniel Quang and Xiaohui Xie. DanQ: A hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences. *Nucleic Acids Research*, 44(11), 2016. ISSN 13624962. doi: 10.1093/nar/gkw226.
12. Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. ISSN 14764687. doi: 10.1038/nature14539.
13. David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587): 484–489, 2016. ISSN 0028-0836. doi: 10.1038/nature16961.
14. Mahmoud Ghandi, Dongwon Lee, Morteza Mohammad-Noori, and Michael A. Beer. Enhanced Regulatory Sequence Prediction Using Gapped k-mer Features. *PLoS Computational Biology*, 10(7), 2014. ISSN 15537358. doi: 10.1371/journal.pcbi.1003711.
15. Ravid Shwartz-Ziv and Naftali Tishby. Opening the Black Box of Deep Neural Networks via Information. Preprint at http://arxiv.org/abs/1703.00810. 2017.
16. Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. Preprint at http://arxiv.org/abs/1610.01644. 2016.
17. Yong Zhang, Tao Liu, Clifford A. Meyer, Jérôme Eeckhoute, David S. Johnson, Bradley E. Bernstein, Chad Nussbaum, Richard M. Myers, Myles Brown, Wei Li, and X. Shirley Shirley. Model-based analysis of ChIP-Seq (MACS). *Genome Biology*, 9(9), 2008. ISSN 14747596. doi: 10.1186/gb-2008-9-9-r137.
18. Robert E. Thurman, Eric Rynes, Richard Humbert, Jeff Vierstra, Matthew T. Maurano, Eric Haugen, Nathan C. Sheffield, Andrew B. Stergachis, Hao Wang, Benjamin Vernot, Kavita Garg, Sam John, Richard Sandstrom, Daniel Bates, Lisa Boatman, Theresa K. Canfield, Morgan Diegel, Douglas Dunn, Abigail K. Ebersol, Tristan Frum, Erika Giste, Audra K. Johnson, Ericka M. Johnson, Tanya Kutyavin, Bryan Lajoie, Bum Kyu Lee, Kristen Lee, Darin London, Dimitra Lotakis, Shane Neph, Fidencio Neri, Eric D. Nguyen, Hongzhu Qu, Alex P. Reynolds, Vaughn Roach, Alexias Safi, Minerva E. Sanchez, Amartya Sanyal, Anthony Shafer, Jeremy M. Simon, Lingyun Song, Shinny Vong, Molly Weaver, Yongqi Yan, Zhancheng Zhang, Zhuzhu Zhang, Boris Lenhard, Muneesh Tewari, Michael O. Dorschner, R. Scott Hansen, Patrick A. Navas, George Stamatoyannopoulos, Vishwanath R. Iyer, Jason D. Lieb, Shamil R. Sunyaev, Joshua M. Akey, Peter J. Sabo, Rajinder Kaul, Terrence S. Furey, Job Dekker, Gregory E. Crawford, and John A. Stamatoyannopoulos. The accessible chromatin landscape of the human genome. *Nature*, 489(7414):75–82, 2012. ISSN 00280836. doi: 10.1038/nature11232.
19. Jesse Davis and Mark Goadrich. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning - ICML '06*, pages 233–240, 2006. ISBN 1595933832. doi: 10.1145/1143844.1143874.
20. Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 7-12:1–9, 2015. ISSN 10636919. doi: 10.1109/CVPR.2015.7298594.
21. J. N.Mark Glover and Stephen C. Harrison. Crystal structure of the heterodimeric bZIP transcription factor c-Fos–c-Jun bound to DNA, 1995. ISSN 00280836.
22. Toshiyuki Shimizu, Atsuki Toumoto, Kentaro Ihara, Masato Shimizu, Yoshimasa Kyogoku, Nobuo Ogawa, Yasuji Oshima, and Toshio Hakoshima. Crystal structure of PHO4 bHLH domain-DNA complex: Flanking base recognition. *EMBO Journal*, 16(15):4689–4697, 1997. ISSN 02614189. doi: 10.1093/emboj/16.15.4689.
23. Xu Min, Wanwen Zeng, Shengquan Chen, Ning Chen, Ting Chen, and Rui Jiang. Predicting enhancers with deep convolutional neural networks. *BMC Bioinformatics*, 18, 2017. ISSN 14712105. doi: 10.1186/s12859-017-1878-3.
24. Sam John, Peter J Sabo, Robert E Thurman, Myong-Hee Sung, Simon C Biddie, Thomas A Johnson, Gordon L Hager, and John A Stamatoyannopoulos. Chromatin accessibility predetermines glucocorticoid receptor binding patterns. *Nat. Genet.*, 43(3):264–268, 2011. ISSN 1061-4036.
25. Shashank Singh, Yang Yang, Barnabas Poczos, and Jian Ma. Predicting Enhancer-Promoter Interaction from Genomic Sequence with Deep Neural Networks. *bioRxiv*, pages 1–12, 2016. doi: 10.1101/085241.
26. Dominic Schmidt, Petra C. Schwalie, Michael D. Wilson, Benoit Ballester, Ângela Gonalves, Claudia Kutter, Gordon D. Brown, Aileen Marshall, Paul Flicek, and Duncan T. Odom. Waves of retrotransposon expansion remodel genome organization and CTCF binding in multiple mammalian lineages. *Cell*, 148(1-2):335–348, 2012. ISSN 00928674. doi: 10.1016/j.cell.2011.11.058.
27. Charles E. Grant, Timothy L. Bailey, and William Stafford Noble. FIMO: Scanning for occurrences of a given motif. *Bioinformatics*, 27(7):1017–1018, 2011.
28. Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. Preprint at http://arxiv.org/abs/1312.6034. *arXiv.org*, 2013.
29. Stephen G. Landt, Georgi K. Marinov, Anshul Kundaje, Pouya Kheradpour, Florencia Pauli, Serafim Batzoglou, Bradley E. Bernstein, Peter Bickel, James B. Brown, Philip Cayting, Yiwen Chen, Gilberto DeSalvo, Charles Epstein, Katherine I. Fisher-Aylor, Ghia Euskirchen, Mark Gerstein, Jason Gertz, Alexander J. Hartemink, Michael M. Hoffman, Vishwanath R. Iyer, Youngsook L. Jung, Subhradip Karmakar, Manolis Kellis, Peter V. Kharchenko, Qunhua Li, Tao Liu, X. Shirley Liu, Lijia Ma, Aleksandar Milosavljevic, Richard M. Myers, Peter J. Park, Michael J. Pazin, Marc D. Perry, Debasish Raha, Timothy E. Reddy, Joel Rozowsky, Noam Shoresh, Arend Sidow, Matthew Slattery, John A. Stamatoyannopoulos, Michael Y. Tolstorukov, Kevin P. White, Simon Xi, Peggy J. Farnham, Jason D. Lieb, Barbara J. Wold, and Michael Snyder. ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia, 2012. ISSN 10889051.
30. Daria Shlyueva, Gerald Stampfel, and Alexander Stark. Transcriptional enhancers: From properties to genome-wide predictions, 2014. ISSN 14710064.
31. Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *Bernoulli*, (1341):1–13, 2009.
32. Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding Neural Networks Through Deep Visualization. Preprint at http://arxiv.org/abs/1506.06579. 2015.
33. Shobhit Gupta, John A. Stamatoyannopoulos, Timothy L. Bailey, and William Stafford Noble. Quantifying similarity between motifs. *Genome Biology*, 8(2), 2007. ISSN 14747596. doi: 10.1186/gb-2007-8-2-r24.
34. Kazutoshi Takahashi and Shinya Yamanaka. Induction of Pluripotent Stem Cells from Mouse Embryonic and Adult Fibroblast Cultures by Defined Factors. *Cell*, 126(4):663–676, 2006. ISSN 00928674. doi: 10.1016/j.cell.2006.07.024.
35. Chin Tong Ong and Victor G. Corces. CTCF: An architectural protein bridging genome topology and function, 2014. ISSN 14710064.
36. Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.*, 45(11):2673–2681, 1997. ISSN 1053587X. doi: 10.1109/78.650093.

DRAFT

37. Aaron R. Quinlan and Ira M. Hall. BEDTools: A flexible suite of utilities for comparing genomic features. *Bioinformatics*, 26(6):841–842, 2010. ISSN 13674803. doi: 10.1093/bioinformatics/btq033.
38. Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735.
39. Geoffrey E. Hinton, Nitish Srivastava, and Kevin Swersky. Lecture 6e- rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, pages 26–31, 2012. doi: https://www.coursera.org/learn/neural-networks/lecture/YQHki/rmsprop-divide-the-gradient-by-a-running-average-of-its-recent-magnitude.
40. Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. pages 1–15, 2014. ISSN 09252312. doi: http://doi.acm.org.ezproxy.lib.ucf.edu/10.1145/1830483.1830503.
41. James T. Robinson, Helga Thorvaldsdóttir, Wendy Winckler, Mitchell Guttman, Eric S. Lander, Gad Getz, and Jill P. Mesirov. Integrative genomics viewer, 2011. ISSN 10870156.
42. Fidel Ramírez, Devon P. Ryan, Björn Grüning, Vivek Bhardwaj, Fabian Kilpert, Andreas S. Richter, Steffen Heyne, Friederike Dündar, and Thomas Manke. deepTools2: a next generation web server for deep-sequencing data analysis. *Nucleic acids research*, 44(W1): W160–W165, 2016. ISSN 13624962. doi: 10.1093/nar/gkw257.

DRAFT