fast decoding cell type-specific transcription factor binding landscape at single-nucleotide resolution

Hongyang Li[1], Yuanfang Guan[1,*]

1. Department of Computational Medicine and Bioinformatics, University of Michigan, 100 Washtenaw Avenue, Ann Arbor, MI 48109, USA

* Corresponding author: gyuanfan@umich.edu

The system configuration to test runtimes

GPU

NVIDIA GeForce GTX TITAN X 12GB

CPU

```
Architecture:           x86_64
CPU op-mode(s):         32-bit, 64-bit
Byte Order:             Little Endian
CPU(s):         8
On-line CPU(s) list: 0-7
Thread(s) per core:  2
Core(s) per socket:  4
Socket(s):      1
NUMA node(s):           1
Vendor ID:              GenuineIntel
CPU family:             6
Model:          158
Model name:             Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz
Stepping:       9
CPU MHz:                800.061
CPU max MHz:            4500.0000
CPU min MHz:            800.0000
BogoMIPS:               8400.00
Virtualization:         VT-x
L1d cache:              32K
L1i cache:      32K
L2 cache:       256K
L3 cache:       8192K
NUMA node0 CPU(s):   0-7
```

Memory

31GB in total

System

Linux version 4.15.0-62-generic (buildd@lcy01-amd64-024) (gcc version 7.4.0 (Ubuntu 7.4.0-1ubuntu1~18.04.1)) #69-Ubuntu SMP Wed Sep 4 20:55:53 UTC 2019

Quantile normalization for new data

Here we use a new cell line, Ag04449, as an example to demonstrate how to perform quantile normalization and generate the average signals for the delta-DNase-seq. The reference genome is GRCh37/hg19. See example code on our GitHub page:
(https://github.com/GuanLab/Leopard#quantile-normalization-for-new-data).

1. Download the Ag04449 DNase-seq bigwig file from the ENCODE-DCC ftp. The Ag04449 has two replicates and we download both of them. It also works if a cell line only has one replica.

2. Download the liver DNase-seq bigwig for our web server. It is used as the reference cell line in Leopard. Other cell lines can also be used as the reference cell line. As long as the same reference cell line is used in model training and prediction, it will not affect the result in most situations.

3. Subsample the reference for quantile normalization
We first subsample a subset from the human genome to estimate the overall distribution, which will save a lot of computing time and memory.

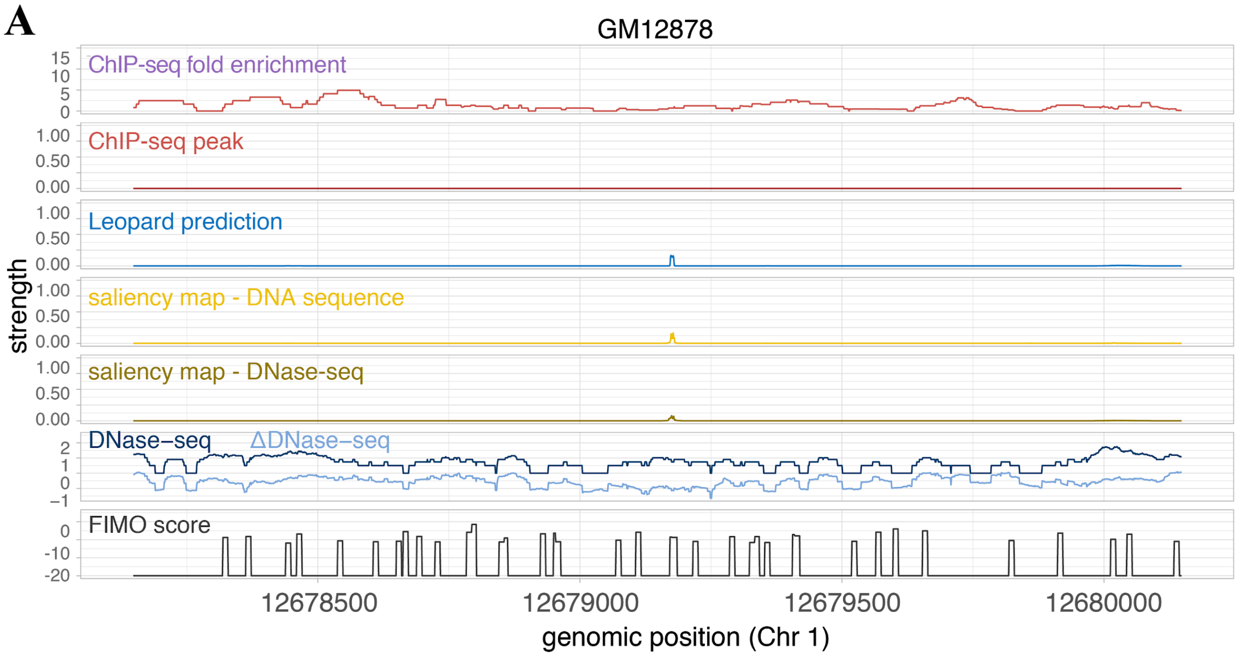
4. Subsample the input files
In this example, the two Ag04449 replicates are used to calculate the average DNase-seq signals. Then we subsample a subset from the same regions in step 3. This subset will be used to estimate the overall distribution of Ag04449 DNase-seq signal.

5. quantile normalization
Based on the subsampled data from the previous two steps, quantile normalize the genome-wide signal of Ag04449 to the reference liver cell line.

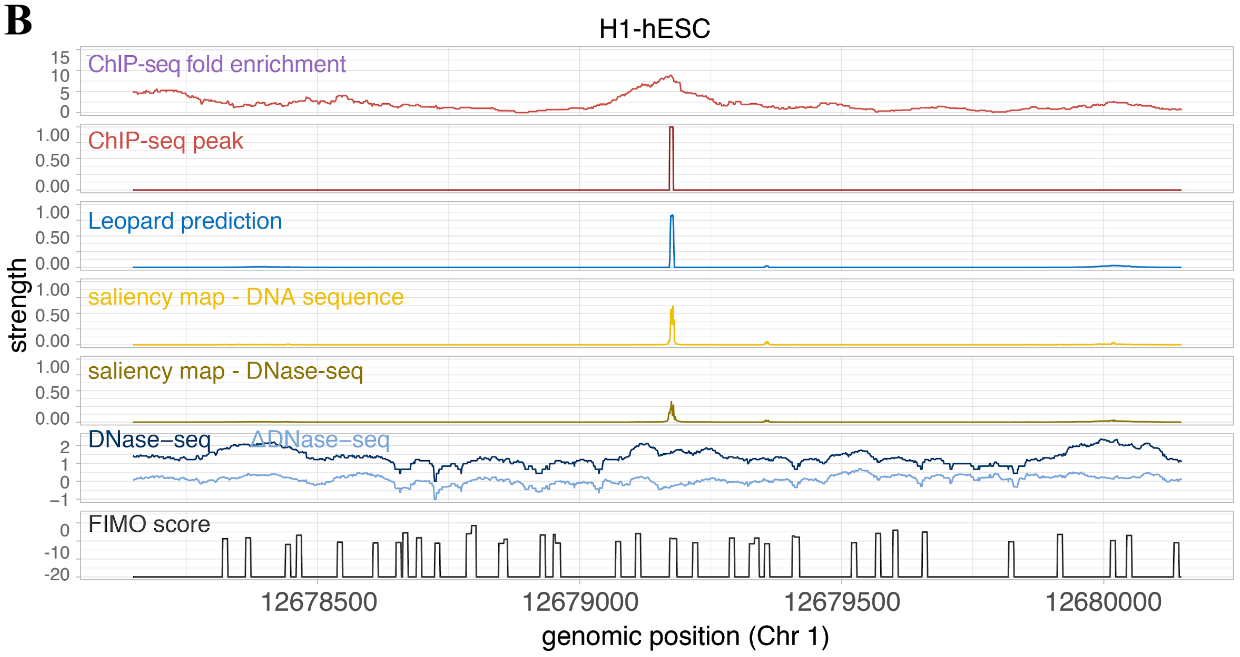
6. calculate the average signals from all cell lines under consideration.
In Leopard, we used all 13 cell lines to calculate the avg.bigwig. Of note, when a new testing cell line comes, we don't need to re-calculate this reference. In general, about 10 cell lines should be enough to generate a robust average signal.

Supplemental Figures

A



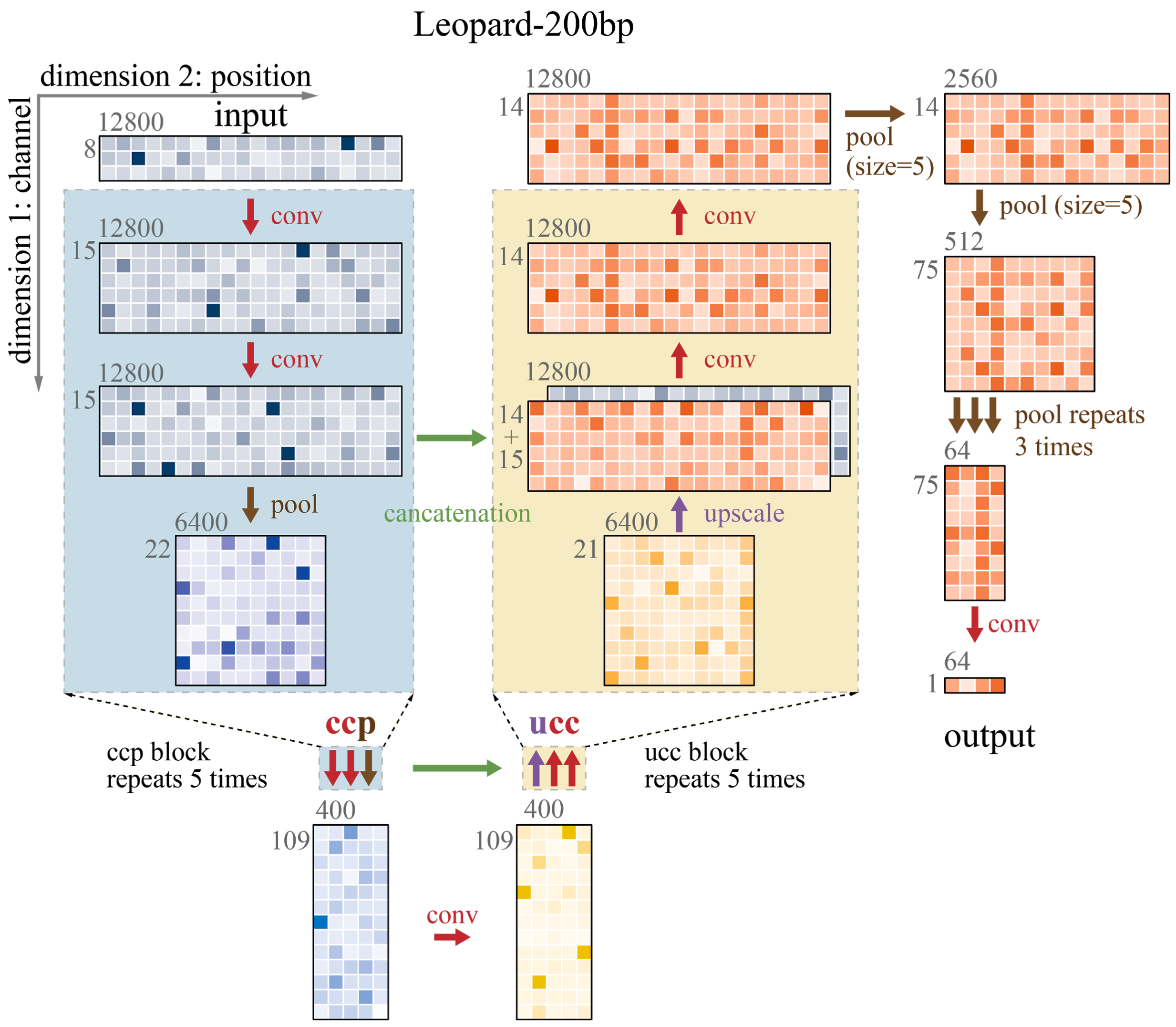
B



Supplemental Fig. S1: Leopard distiguishes cell type-specific transcription binding and non-binding events at single-nucleotide resolution.

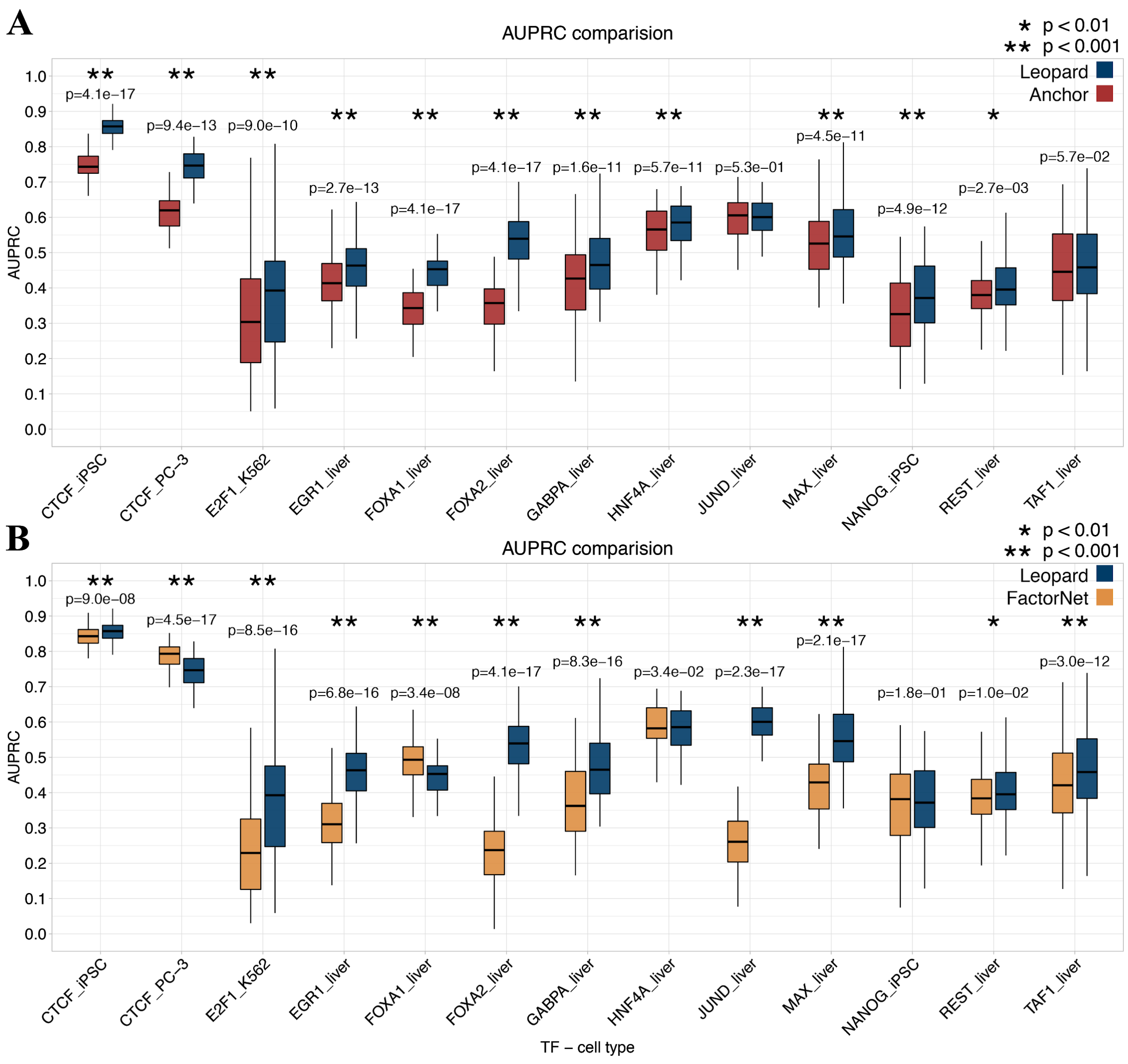
An example 2000bp segment is shown to demonstrate Leopard prediction results across three cell types: **A,** GM12878 **B,** H1-hESC and liver (Fig. 2E). This segment contains successive signals between genomic positions 12,678,147 and 12,680,147 of Chr 1 from the JUND binding profiles. The first row is the original ChIP-seq fold enrichment generated through the standard

ENCODE analysis pipeline. The second row is the high-resolution ChIP-seq peaks created by the GEM peak finder. Leopard generates single-nucleotide predictions and precisely provides the potential binding sites in H1-hESC and liver cell types. The saliency map indicates positions contributing to the predictions. The corresponding DNase-seq and ΔDNase-seq signals, and the sequence-based motif scan scores using FIMO are also shown here for comparison. Of note, no JUND binding events were observed by ChIP-seq experiments in GM12878 cell type. Leopard successfully depicted the differences across cell types and made very tiny-peak predictions only in GM12878.



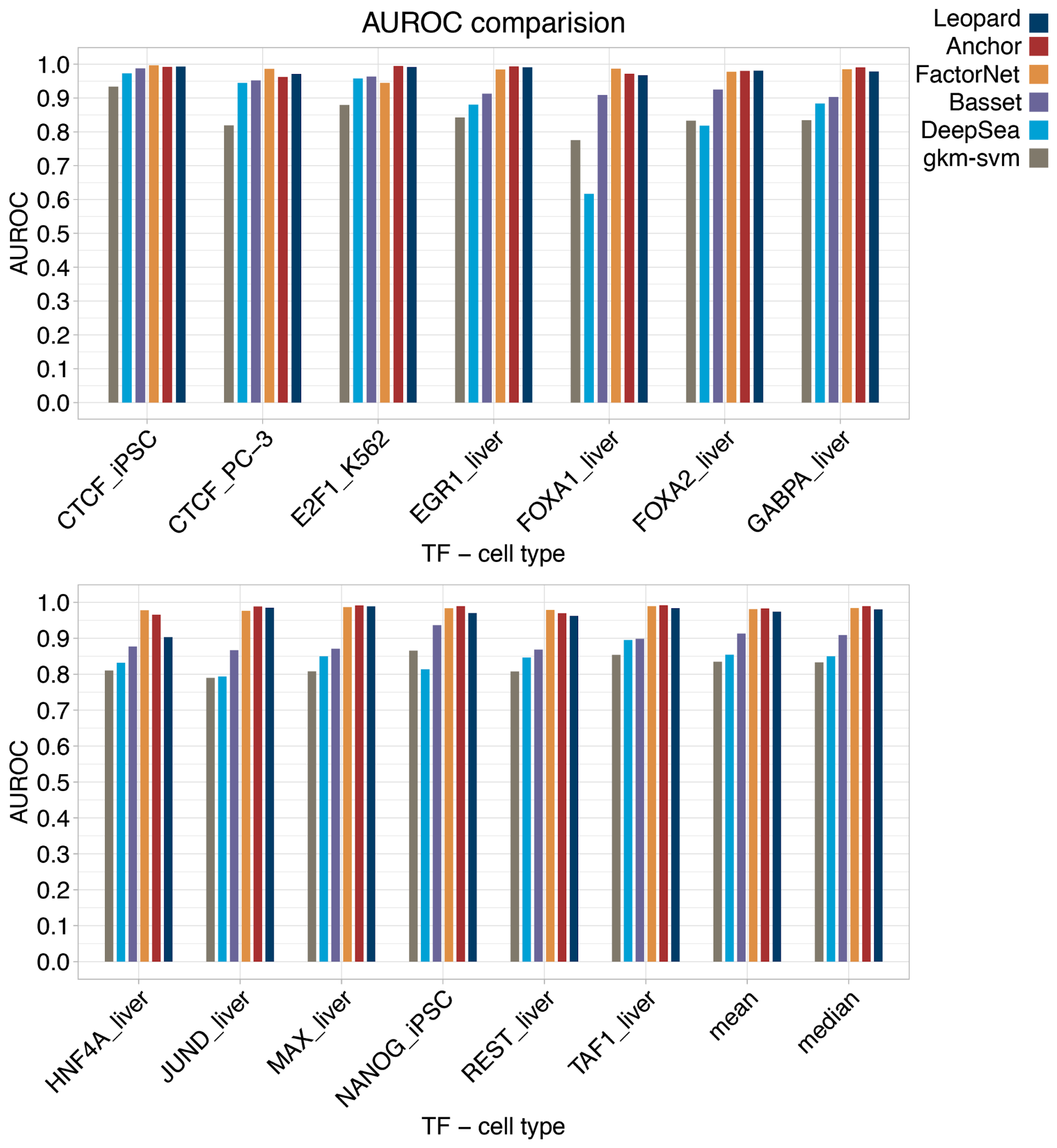
Supplemental Fig. S2: The architecture of Leopard-200bp designed for predicting genome-wide TF binding sites at 200bp resolution.

The encoder (blue) and decoder (yellow) components of Leopard-200bp are similar to those of Leopard, except for the input length was adjusted from 10240 to 12800. We made this adjusting because 12800 (instead of 10240) can be divided by 200 so that the output is a 1-by-64 array corresponding to predictions at 200bp resolution. This is accomplished by multiple max-pooling layers (brown arrows on the right) gradually reducing the size from 12800 to 64.



Supplemental Fig. S3: The statistical comparison of Leopard with the state of the art.
For each testing TF-cell type pair, we randomly sampled 100 segments with length of 100kbp and calculated 100 prediction AUPRCs. The paired Wilcoxson signed rank test was performed.

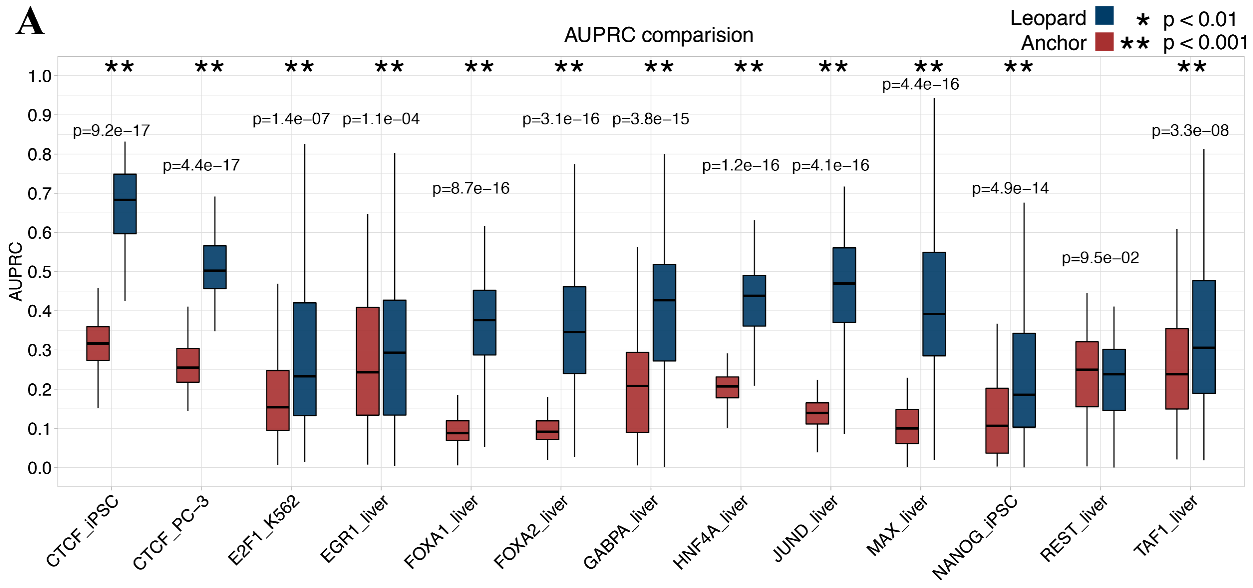
Leopard significantly outperformed **A,** Anchor and **B,** FactorNet in the majority of testing TF-cell type pairs. The significant differences are labeled with a single asterisk (p-value < 0.01) or double asterisks (p-value < 0.001).



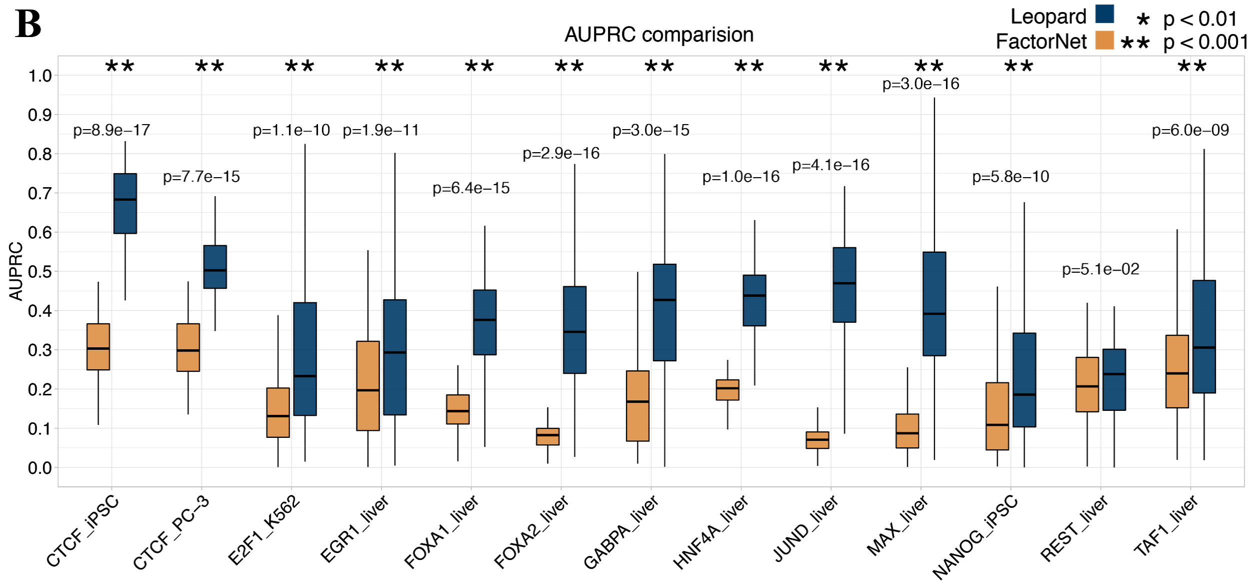
Supplemental Fig. S4: AUROC comparison with state-of-the-art methods

Leopard was benchmarked with Anchor, FactorNet, Basset, DeepSea, gkm-svm on testing data from the ENCODE-DREAM *in vivo* Transcription Factor Binding Site Prediction Challenge. For the top three methods (Leopard, Anchor, FactorNet), the AUROCs of these methods are comparable and no significant differences are observed using the Wilcoxon Signed Rank test (p-value = 0.0215 between Leopard and Anchor; p-value=0.244 between Leopard and FactorNet).

A

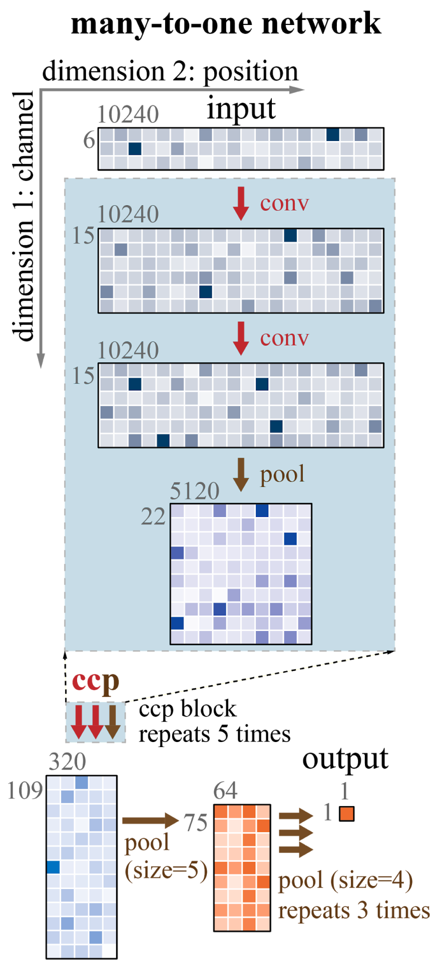


B



Supplemental Fig. S5: Comparison of Leopard, Anchor and FactorNet at 1bp resolution.

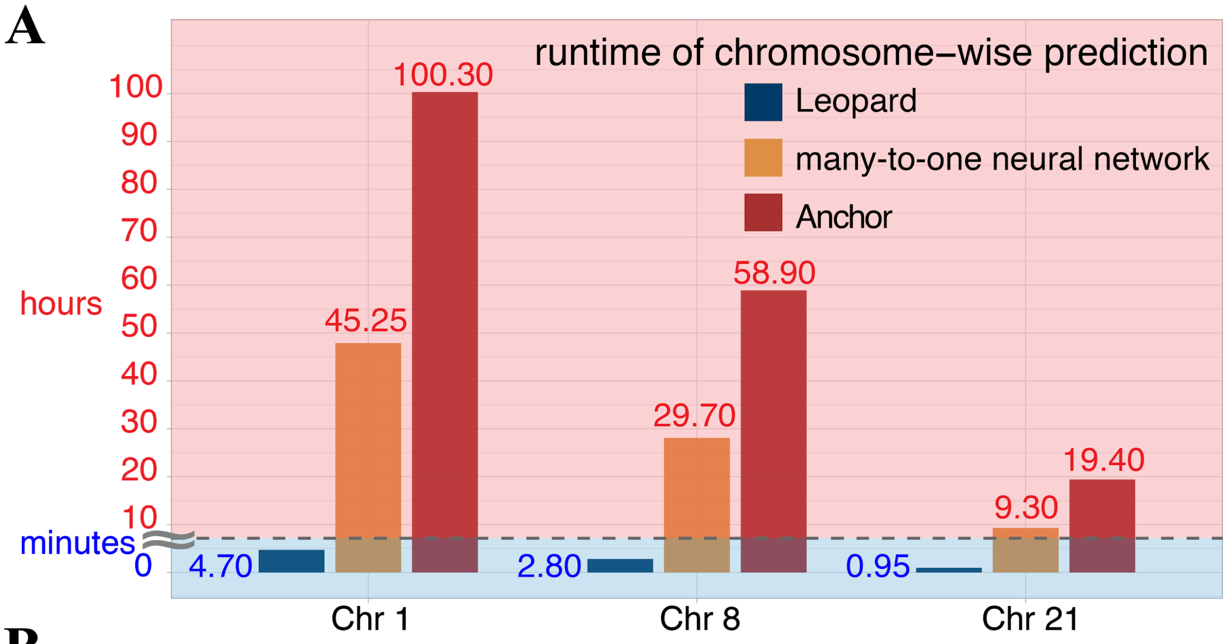
Since Anchor and FactorNet only provide predictions at the 200bp interval, their predictions for each 200bp interval were repeated 200 times to generate the 1bp predictions. Leopard demonstrates significantly higher AUPRCs than Anchor and FactorNet in the 13 testing TF-cell type pairs.



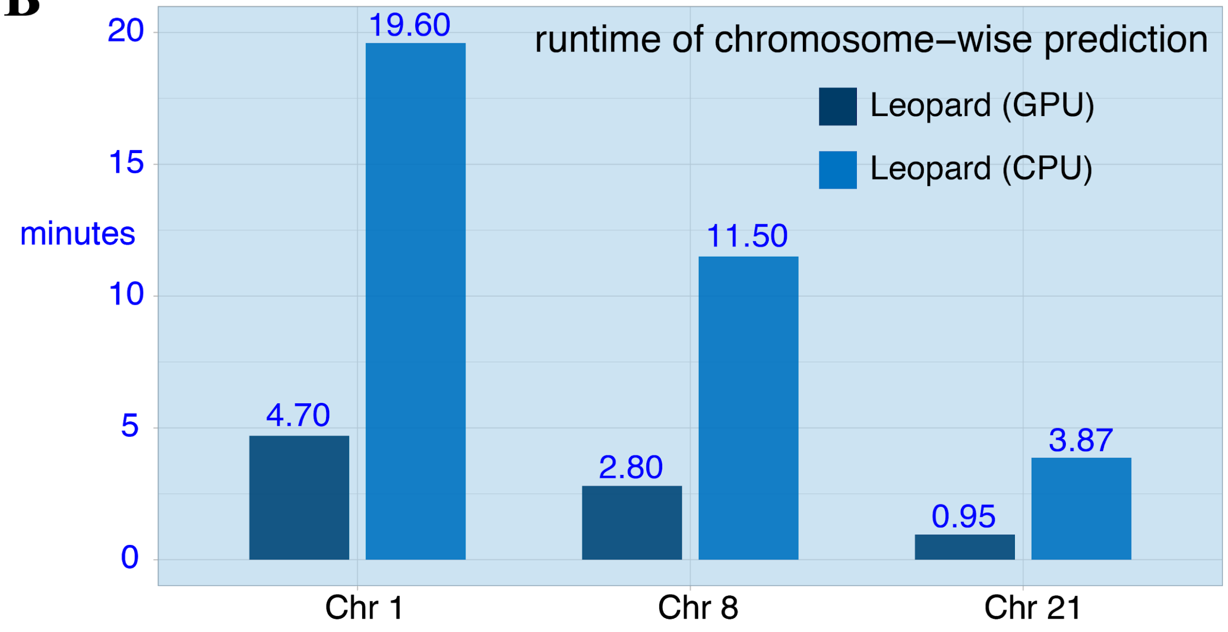
Supplemental Fig. S6: The architecture of the many-to-one model represents a type of convolutional neural network structure commonly used in previous methods.

Compared with Leopard, the many-to-one model only has the encoder (blue) component without the decoder (yellow), representing the common type of convolutional neural network model used in previous methods. At the end of the encoder, multiple max-pooling layers (brown arrows on the right) were used to gradually reduce the size of the output. This type of model is therefore called “many-to-one”, since the input size is 10240 (many) and the final output is only 1 value (one).

A

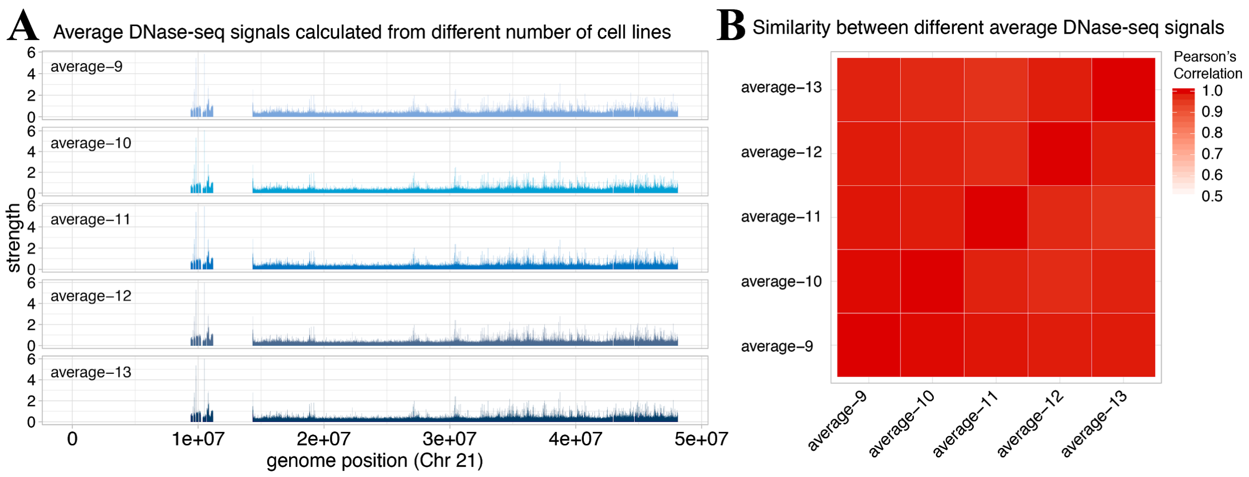


B



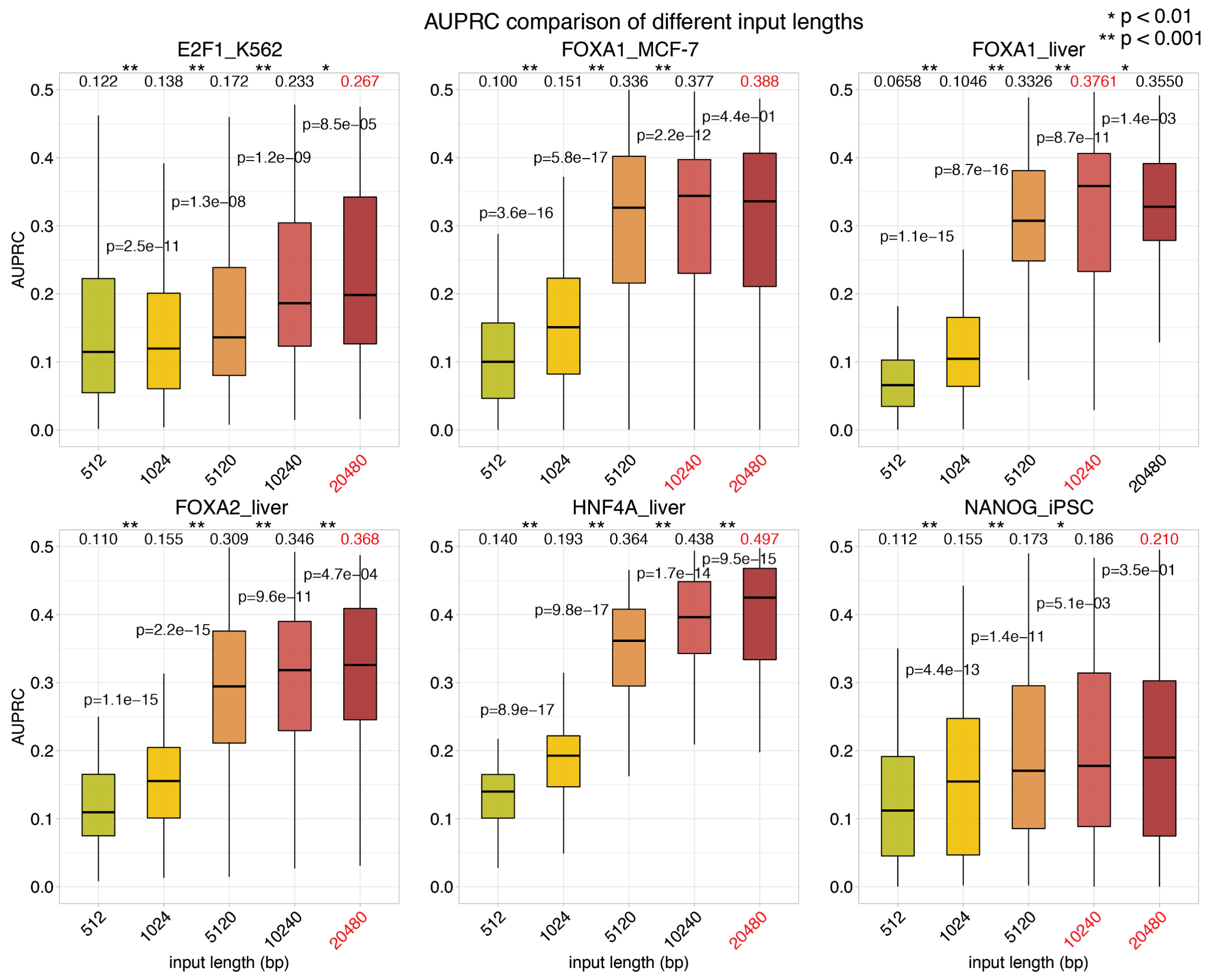
Supplemental Fig. S7: The runtimes of chromosome-wise predictions of Leopard and other state-of-the-art methods.

The runtime was tested for predicting the HNF4A binding profiles in the liver cell using different methods. **A,** The runtime was tested for chromosome-wise predictions on Chr 1, Chr 8 and Chr 21. Notably, the runtimes of bars within the blue background are in the unit of minute, whereas the runtimes in the red background are in the unit of hour. For example on Chr 8, it only takes 2.8 minutes for Leopard to finish predictions whereas it requires 29.70 and 58.90 hours for the many-to-one neural network model and the Anchor method using XGBoost. Leopard is therefore estimated to be 600 and 1200 times faster than the many-to-one model and Anchor on average. **B,** Leopard is flexible with both GPU and CPU settings. When running Leopard on CPU, the prediction runtimes are acceptable on the scale of minutes. The runtime of CPU was tested on a standard computer with eight CPU cores.



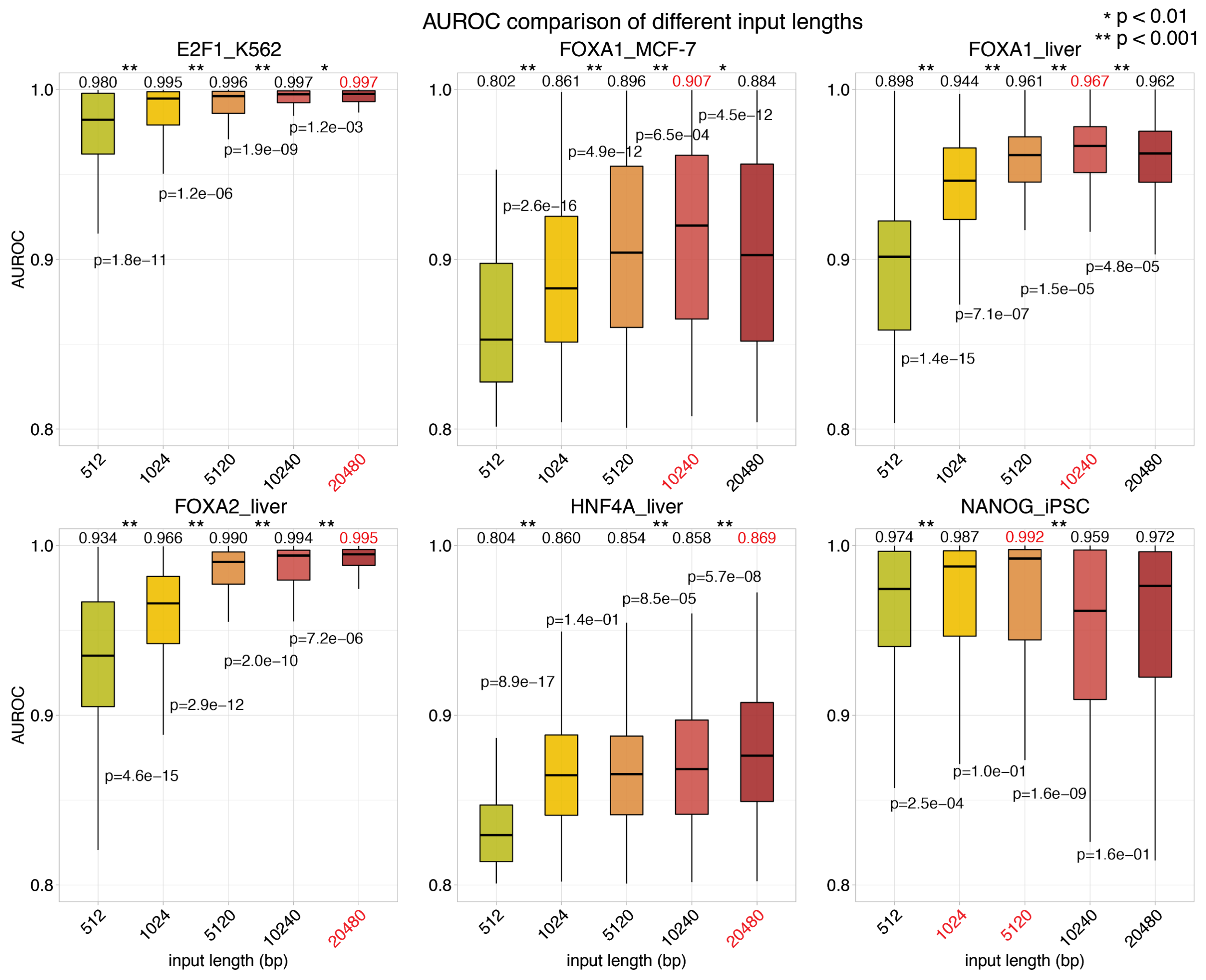
Supplemental Fig. S8: Calculating the average reference using different numbers of cell types

a, In Leopard, we used all 13 cell types to calculate the average. We also randomly selected 9, 10, 11, 12 cell types to calculate the average. The overall signals are similar and here Chr 21 is used as an example. **b,** The pairwise Pearson's correlations among the five average references. All the correlation values are above 0.95.



Supplemental Fig. S9: AUPRC Comparison of models with different input lengths.

The predictive performance of models with five different input lengths (512bp, 1024bp, 5120bp, 10240bp, 20480bp) are evaluated in six TF-cell type pairs. Longer inputs generally lead to significantly better performance.



Supplemental Fig. S10: AUROC Comparison of models with different input lengths.

The predictive performance of models with five different input lengths (512bp, 1024bp, 5120bp, 10240bp, 20480bp) are evaluated in six TF-cell type pairs. Longer inputs generally lead to significantly better performance.

Supplemental Table Legends

Supplemental Table S1. The dimensions and kernel sizes of each layer in Leopard architecture. Each row describes the information of a layer in our neural network model. Each layer has two dimensions, where the first dimension is the length along the genomic coordinate and the second dimension is the channel.

Supplemental Table S2. The training and testing partition of 51 ChIP-seq data used in this study. Each row describes a TF and each column describes a cell type. The training and testing data are shown in blue and red, respectively.

Supplemental Table S3. The AUPRC baselines of random predictions and the predictions by overlapping DNase-seq signals with FIMO motif scanning hits.
For each pair, we calculated the baseline of random predictions, which were also the percentage of TF binding sites. We also calculated a more stringent baseline by overlapping DNase-seq signals with FIMO motif scanning hits.

Supplemental Table S4. The partial AUPRCs and other metrics at different recall cutoffs for Leopard predictions of the ChIP-seq peaks.
To fully understand the performance of Leopard predictions on ChIP-seq data, we calculated the partial AUPRCs and related metrics using different recall cutoffs at 0.01, 0.05, 0.10, 0.50, 0.80 and 0.90.

Supplemental Table S5. The partial AUROCs and other metrics at different FPR cutoffs for Leopard predictions of the ChIP-seq peaks.
To fully understand the performance of Leopard predictions on ChIP-seq data, we calculated the partial AUROCs and related metrics using different FPR cutoffs at 0.0001, 0.001, 0.005, 0.01, 0.05 and 0.1.

Supplemental Table S6. The partial AUPRCs and other metrics at different recall cutoffs for Leopard predictions of the ChIP-exo peaks.
To fully understand the performance of Leopard predictions on ChIP-exo data, we calculated the partial AUPRCs and related metrics using different recall cutoffs at 0.01, 0.05, 0.10, 0.50, 0.80 and 0.90.

Supplemental Table S7. The partial AUROCs and other metrics at different FPR cutoffs for Leopard predictions of the ChIP-exo peaks.

To fully understand the performance of Leopard predictions on ChIP-exo data, we calculated the partial AUROCs and related metrics using different FPR cutoffs at 0.0001, 0.001, 0.005, 0.01, 0.05 and 0.1.

Supplemental Table S8. The training and testing partition of 56 ChIP-seq data used in the ENCODE-DREAM Challenge.
We summarized the training - testing partition of ChIP-seq data in the ENCODE-DREAM Challenge. The training and testing data are shown in blue and red, respectively.

Supplemental Table S9. The accession numbers of sequence alignments of ChIP-seq reads downloaded from the ENCODE Project.
We listed the accession numbers of ChIP-seq data from the ENCODE Project dataportal. We downloaded the sequence alignment files based on these accession numbers.