Fourier-transform-based attribution priors improve the interpretability and stability of deep learning models for genomics: Supplementary Methods

All code is available at `https://github.com/amtseng/fourier_attribution_priors`

1 Training data preparation

1.1 SPI1 and GATA2 TF ChIP-seq

For these two transcription factors (TFs), we obtain data accessible through the ENCODE portal (`https://encodeproject.org/`), which has been processed using the ENCODE ChIP-seq pipeline [1, 2]. We select every TF ChIP-seq experiment with SPI1 or GATA2 as a target, satisfying the following conditions:

1. Experiment is of "released" status
2. Experiment has available unfiltered read BAMs aligned to hg38
3. Experiment has available BEDs of called peaks and IDR-filtered peaks aligned to hg38
4. Experiment has a matched control ChIP experiment with unfiltered read BAMs aligned to hg38
5. Cell type utilized in assay is not genetically modified

Based on these filtering criteria as of 4 Oct 2019, the ENCODE experiment IDs for SPI1 are ENCSR000BGQ, ENCSR000BUW, ENCSR000BIJ, ENCSR000BGW. The ENCODE experiment IDs for GATA2 are ENCSR000EVW, ENCSR000EWG, ENCSR000EYB. For each experiment, we obtain the unfiltered alignments, called peaks, and IDR-thresholded peaks, as available on the ENCODE portal (`https://encodeproject.org/`). For each TF, each experiment constitutes an output in the multi-task prediction models.

1.2 Nanog/Oct4/Sox2 TF ChIP-seq

Nanog, Oct4, and Sox2 TF ChIP-seq experiments in mouse ESCs were performed by Avsec et al. [3]. We use the stranded BigWig tracks (5' read counts) and IDR-thresholded called peaks as prepared by the authors. This constitutes three different experiments (i.e. prediction tasks). These reads and peaks are aligned to mm10.

1.3 K562 DNase-seq

We utilize a single DNase-seq experiment available through the ENCODE portal (`https://encodeproject.org/`): ENCSR000EOT [1, 2]. For this experiment, we obtain the unfiltered alignment BAMs, the set of called peaks that pass the ENCODE IDR filter, and the set of called peaks that did not pass the IDR filter. These files are obtained using the ENCODE ATAC-seq pipeline [4]. These reads and peaks are aligned to hg38.

Unlike TF ChIP-seq experiments, which have a matched control experiment, we perform bias correction by utilizing a control track that encodes the innate preferences of DNase [5]. We use the reads generated using this procedure, available as SRR1565781. These reads are processed into BigWigs using the ENCODE ATAC-seq pipeline into a single unstranded BigWig [4].

1.4 Reference genomes

We obtain reference genome Fasta files and chromosome sizes from the UCSC genome browser downloads, for hg38 and mm10.

1.5 Binary dataset preparation

The IDR-thresholded peaks are considered to be the high-confidence peaks. For the SPI1 and GATA2 TF ChIP-seq datasets, the 150,000 highest-scoring called peaks that do not overlap with an IDR-thresholded peak are considered ambiguous peaks, obtained using BEDtools v2.25.0 [6]. For the K562 DNase-seq dataset, the set of ambiguous peaks is generated by the ENCODE ATAC-seq pipeline [5]. For the Nanog/Oct4/Sox2 TF ChIP-seq datasets, there are no available ambiguous peak sets.

For all binary models, labels are generated by splitting the entire genome into 200 bp consecutive windows. A window is given a positive label if at least 100 bp of the window overlaps with a high-confidence peak, an ambiguous label if at least 100 bp of it overlaps with an ambiguous peak, and a negative label if neither of these two cases hold. For the TF ChIP-seq binary models, which have multiple tasks, the label generation procedure is performed independently for each task (i.e. each window will have an associated label for each task). These windows are then padded with 400 bp of context sequence on either side to form the final 1000 bp input to the network. This binary label generation was performed using seqdataloader [7].

1.6 Profile dataset preparation

For the SPI1 and GATA2 TF ChIP-seq datasets and the K562 DNase-seq dataset, we first merge together the unfiltered reads of all biological replicates in each experiment. For each experiment, we filter the merged reads using samtools v1.2 with flag 780, and keep only reads with quality at least 30 [8]. This is nearly identical to the ENCODE TF ChIP-seq and ATAC-seq pipelines [1, 4], except this retains duplicate reads, which may be useful in the prediction of profile shapes.

Using bedGraphToBigWig v4 [9], we convert these reads into BigWigs of 5' counts. For TF ChIP-seq datasets, the BigWigs are split into positive and negative strands, but for the K562 DNase dataset, the BigWig is unstranded. This gives each experiment a pair of—or a single—BigWig track(s) of 5' counts.

The profile 5'-count BigWigs for the Nanog/Oct4/Sox2 dataset were obtained from Avsec et al. [3] and are used as-is.

2 Model training

For both model architectures, we use a learning rate of 0.001. We train for a maximum of 20 epochs with early stopping (requiring an improvement in the validation loss by at least 0.001 over the last 3 epoch deltas). For binary models, we use a batch size of 128. For profile models, due to the larger model size, we use a batch size of 64. Batch size, learning rate, and early stopping criteria were selected by choosing values in previous works on similar architectures, and verifying that validation performance was comparable.

We also utilize reverse-complement augmentation in each batch, thereby effectively doubling the batch size.

Binary models learn a set of binary labels from a 1 kb input sequence. Profile models learn a set of 1 kb read profiles from a 1346 bp input sequence, as well as a set of scalar total read counts.

Models were trained using PyTorch 1.3.0, on Google Cloud using n1-standard-8 instances, each with an NVIDIA Tesla P100 GPU.

2.1 Training binary models

The binary model architecture consists of three consecutive 1D convolutions on the one-hot encoded DNA sequence. The convolutional layers have filter sizes of 15, 15, and 13, respectively (stride of 1). We do not use padding on the convolutional layers. For TF ChIP-seq datasets, we use 64 filters in each layer; for K562 DNase-seq models, we use 256. The convolutional layers have ReLU activations and batch normalization. The result of the convolutions are fed into a max-pooling layer of size 40 and stride 40. The max-pooling output is fed into two consecutive fully connected layers of size 50 and 15, respectively, both with ReLU activations. Finally, the result is passed to a final dense layer that outputs a sigmoid prediction. Each dense layer is also batch normalized.

The loss function is binary cross-entropy, averaged across the different tasks. The cross-entropy loss for the positive and negative classes are averaged in each batch. Each batch consists of genome bins where at least one task has a positive label, and an equal number of genome bins where no task had a positive label.

In each epoch, the network sees all genome bins where at least one task has a positive label. The "negative" samples are randomly subsampled at the beginning of each epoch.

2.2 Training profile models

The profile model architecture, adapted from Avsec et al. [3], has a profile output and a total counts output for each task. The architecture consists of seven consecutive 1D dilated convolutions on the one-hot encoded DNA sequence. The first dilated convolutional layer has a filter size of 21, and the subsequent layers have filter sizes of 3 (all have stride of 1). For TF ChIP-seq datasets, we use 64 filters; for K562 DNase-seq models, we use 256. Dilation size is 1 for the first layer (i.e. no dilation), and increases by powers of two in subsequent layers. The dilated convolutional layers have

ReLU activations. These dilated convolutional layers have summed residual connections, where the input to each layer is the sum of the outputs of all previous layers. Due to these summed connections, we utilize zero padding in these layers to ensure that the output of each layer is the same size. At the end, the final output of these dilated convolutional layers is cropped, cutting off any regions whose receptive field included a padded zero.

To compute the profile prediction, the last dilated convolution output is fed to another convolutional layer with kernel size 75 (stride 1) with no padding and no activation function (the output of this layer is length 1000). Finally, this result is then stacked with the control profile tracks, and fed to a final convolutional filter of size 1, such that the filter operates on one base of the logits and control profiles at a time. This constitutes the predicted profile logits.

To compute the total read count prediction, the last dilated convolution output is fed to a global average pooling layer, then to a dense layer. The result is concatenated with the total read counts in the control experiment, and fed through a final dense layer that predicts the log of the total read counts.

To compute the profile output and profile loss, the predicted profile logits are converted into probabilities by passing them through a softmax along the profile prediction dimension. This gives the predicted profile shape. For each track, these probabilities, along with the true read counts over the region, define a multinomial distribution, where each base is a bucket. The profile loss is the (negative) log probability of observing the true reads counts over this distribution, where the post-softmax predicted probabilities define the likelihood of a true read falling in each base. The results over all strands and tasks are averaged.

To compute the total counts loss, the predicted counts are treated as log counts, and the counts loss is simply the mean squared error of the log total counts, averaged over all strands and tasks (with a pseudocount of 1 for numerical stability).

When training, the profile loss and counts loss are given a weight of 1 and 20, respectively.

A positive example for a profile model is an input sequence and target profile track set centered at an IDR-thresholded peak summit (for any of the tasks). In each epoch, the network sees all IDR-thresholded peaks (the union over all tasks). Each batch also consists of an equal number of non-peak sequences/profiles, sampled uniformly at random from the genome every time a batch is generated. This implies that a sampled sequence intended for the negative set may overlap a peak, but note that not only is this unlikely, it also does not affect the correctness of any labels passed to the model. The peak sequences are also randomly jittered up to 128 bp from the summit in either direction, to augment the set of positive examples; this is done independently in each batch. For the K562 profile models, we do not train with a negative set due to time efficiency, as we anecdotally found that the profiles with random jitters were sufficient to yield good performance.

Our profile models also utilize control profiles for bias correction. For the SPI1, GATA2, and K562 models, these are matched controls. For the Nanog/Oct4/Sox2 profile models, the control is identical and multiplexed across all tasks.

Our TF ChIP-seq profile models (i.e. for SPI1, GATA2, and Nanog/Oct4/Sox2) are stranded, and predicted and control profiles have positive and negative strands. Our K562 DNase-seq profile models, however, are unstranded, and the predicted and control profiles are summed across both strands.

2.3 Fourier-based prior loss

In each batch during training, we only compute the Fourier-based prior loss for positive examples (i.e. binary examples where at least one task had a positive label, or profile examples originating from a peak). The Fourier-based prior loss is computed independently for each positive input sequence in the batch, and the loss is averaged.

The attributions $g(f, x)$ are computed for each positive example, resulting in a vector of the same length N as the input sequence. In our models, $g(f, x)$ is based on the element-wise product of the input one-hot encoded sequence and the gradient of the output logits with respect to the input (as described in Shrikumar et al. [10]). This element-wise product is then summed over the base dimension, yielding a single score for each position in a given sequence. For binary models, we simply multiply the input sequence and the gradient of the binary logits with respect to the input sequence. For profile models, analogous to Avsec et al. [3], we use the profile prediction logits (pre-softmax), weighted by the post-softmax probabilities, and summed across the entire profile; these logit gradients are then multiplied by the input sequence to yield $g(f, x)$. For models with multiple tasks or strands, these logits are summed across these dimensions; thus, every positive training example will have a vector of attributions $g(f, x)$ that is the same length as the input sequence.

Note that g, our method of computing attribution priors, needs to be differentiable in order to include it in the attribution prior loss. Fortunately, for typical deep learning models (which are differentiable everywhere by construction), g is also differentiable everywhere for most forms of backpropagation-based attribution methods (e.g. input gradients).

To compute the attribution prior loss, We take the absolute value of these attributions, and smooth them with a Gaussian kernel of standard deviation equal to 3 bp, cut to 1 standard deviation on each side (that is, the kernel has a width of 7 bp). The absolute-value and smoothing operations are intended to aid the prior in gracefully considering motifs with degenerate bases or variants (e.g. negative-scoring bases). Let the resulting smoothed attribution vector be $g^s(f, x)$. We then compute the discrete Fourier transform and recover the magnitudes $m^{(DC)}$ of the positive Fourier frequencies:

$$m^{(DC)} = FFT(g^s(f, x))$$

Note that $m^{(DC)}$ is a vector of length $\frac{N}{2} + 1$. We discard the component $m_0^{(DC)}$ which corresponds to "DC" (i.e. the average value of the attributions) to obtain m. m is a vector of length $L = \frac{N}{2}$. We ℓ_1-normalize the magnitudes m:

$$\hat{m} = \frac{m}{\sum_{i=1}^{L} m_i}$$

Finally, we compute the attribution prior loss as the sum of the high-frequency normalized magnitudes:

$$w_i = \begin{cases} 1 & i \leq T \\ \frac{1}{1+(i-T)^s} & i > T \end{cases}$$

$$L_p(g(f, x)) = 1 - \sum_{i=1}^{L} w_i \hat{m}_i$$

We utilize a soft cut-off, with $s = 0.2$. This cut-off was selected by visualizing the graph of $h(x) = \frac{1}{1+x^s}$ and selecting a reasonable s such that $h(x)$ decays gracefully over a span of roughly 50 bp (Supplementary Figure S1).

For a signal of length N, a rectangular pulse of size p has a discrete Fourier transform of a sinc function with zeros every integer multiple of $\frac{N}{p}$. For binary models, the cut-off T is selected to be 150 (in terms of frequency index), and for profile models, the cut-off is 200. In both cases, this corresponds to a motif length (i.e. pulse) of 6–7 bp (the input sequences to the binary and profile models have different lengths, and hence a different frequency index corresponds to a pulse of the same size). Note that this frequency threshold can be increased accordingly for datasets/tasks in which motifs are expected to be shorter than 6–7 bp, or vice versa.

When training with the Fourier-based prior, the prior loss is weighted by 1 for binary models. For profile models, we train a model first without the Fourier-based prior, then we take half of the validation loss (rounded to the nearest multiple of 5) after the model converges to be the weight of the prior loss (Supplementary Figure S4).

Aside on the periodicity of the discrete Fourier transform: Because the discrete Fourier transform implicitly constructs an infinitely long periodic signal from its input (i.e. by concatenating it end-to-end *ad infinitum*), there can sometimes be artifacts in the frequency domain. Fortunately, however, these issues are not typically present in our application, as the ends of the input sequence are generally devoid of any motifs or other informative features (and ergo the importance scores at the ends of the sequences are zero).

2.4 Peak subsampling

In some of our downstream analyses, we train models with only a subset of the dataset (e.g. 1% of the training set). To subsample our dataset, we limit the set of IDR-thresholded peaks to only the top 1% by signal strength. We do this by ranking all IDR-thresholded peaks across all tasks in descending order by signal strength, removing duplicate peaks (i.e. perfect overlaps), and retaining only the top 1% by count. This keeps only the 1% strongest/most confident peaks. For binary models, we recreate the training set from this new set of limited peaks. Negative bins are selected during training as usual. For profile models, we simply train with this smaller set of peaks as the positive set. Negative examples training are selected genome-wide, as usual.

For the Nanog/Oct4/Sox2 profile models trained on just 1% of the training data, we train for 80 epochs instead of the usual 20 (both with or without the Fourier-based prior).

2.5 Training logistics

In all models, we reserve chr1 as the test set, reserve chr8 and chr10 as the validation set, and partition all other chromosomes into the training set (for hg38 or mm10). Evaluation of a model on the validation and test set proceeds identically to the training set in terms of the selection of positive and negative examples (i.e. we utilize all positive examples in the validation/test set, and an equally sized random sample of negative examples).

To evaluate binary model performance, we examine the validation loss, and the test-set accuracy, auROC, and auPRC. These metrics are computed using balanced positive and negative classes. Because the auPRC is sensitive to class imbalance, we also compute an "estimated test auPRC", which estimates the true auPRC on the full test set as if the negative examples were not subsampled to achieve a balanced set. We use this estimate rather than computing the auPRC on the full test set for computational efficiency. This estimate is calculated by artificially inflating the false positive rate, with the assumption that the negatives already present are representative of the full distribution of negatives.

To evaluate profile model performance, we examine the profile loss (i.e. the negative log-likelihood of the profiles).

For each dataset and model architecture, we train models over 30 different random initializations. In our downstream analyses (unless otherwise stated), we always consider the model with the best validation loss over all random initializations and epochs. For profile models, we compare the profile validation loss, rather than the aggregate loss of the profiles and counts outputs.

2.6 L2-regularization models

We train SPI1 binary models with L2-regularization (without the Fourier-based prior), using an added L2-norm loss, consisting of the L2-norm of all trainable parameters in the network.

We tune the L2-norm loss weight over 25 logarithmically sampled random weights from 10^{-8} to 10^{2}, eventually settling on the optimal weight of 0.0001, which yielded the lowest validation loss. Using this optimal L2 loss weight, we train 20 random initializations of SPI1 binary models. In all analyses, we use the model/epoch with the best validation loss.

2.7 Smoothness prior and sparsity prior

We train SPI1 binary models with the smoothness prior or the sparsity prior (without L2-regularization or the Fourier-based prior), as described in Erion et al. [11]. Let $g(f, x)$ be the vector of attributions, and N the vector's length, as described in Sec. 2.3. We take the absolute value of the attributions to obtain the non-negative vector $g^a(f, x)$. Let $g^a(f, x)_i$ be the ith element of $g^a(f, x)$.

Although the smoothness prior is defined in Erion et al. [11] to penalize differences in attributions between adjacent pixels in images, we adapt it for sequence inputs as follows:

$$L_p(g(f, x)) = \sum_{i=1}^{N-1} |g^a(f, x)_i - g^a(f, x)_{i+1}|$$

The sparsity prior is defined as:

$$L_p = -\frac{\sum_{i=1}^{N}\sum_{j=1}^{N} |g^a(f, x)_i - g^a(f, x)_j|}{N \sum_{i=1}^{N} g^a(f, x)_i}$$

For each prior, we train models over 15 random initializations. We select the prior loss weight λ so that the ratio of L_c to λL_p is roughly the same as the ratio for a trained SPI1 binary model using the Fourier-based prior (L_c is the model's correctness loss, which in this case is binary cross entropy). This ensures that the relative magnitudes of L_c and λL_p are roughly the same. Using this method, we select a prior loss weight of 20000 for the smoothness prior, and 0.01 for the sparsity prior. We pick the models with the best validation loss for downstream analyses where we consider models without a tuned prior loss weight.

We then further tune the prior loss weight for the smoothness and sparsity priors. For the smoothness prior, we tune over 15 logarithmically sampled random weights from 10^{-3} to 10^{7}. For the sparsity prior, we tune over 15 logarithmically sampled random weights from 10^{-4} to 10^{2}. The best prior weights were roughly 5000 for the smoothness prior, and 0.003 for the sparsity prior. We pick the models with the best validation loss for downstream analyses where we examine models after tuning the prior weight.

2.8 Models on simulated sequences

We train single-task SPI1 binary models with simulated DNA sequences for the positive and negative labels. We also only train for 5 epochs, where each epoch has the same number of positive and negative examples as a single-task SPI1 binary model trained on ENCSR000BGQ. All other training details are identical to the SPI1 binary models on real experimental data.

To create the input sequence for a negative label, we synthesize a sequence where every base is sampled independently and uniformly from A, C, G, and T. To create the input sequence for a positive label, we similarly sample a sequence from a uniform distribution of bases, and subsequently place a single instance of the SPI1 motif in the center. The placed motif is sampled from a Position Frequency Matrix (PFM) that was generated using HOMER 2 [12] on the set of IDR-thresholded peaks for SPI1, aggregated over all 4 ENCODE experiments (Supplementary Figure S21). Specifically, we used findMotifsGenome.pl in hg38, with a length of 12 and size of 200. We keep only the top motif. We then trim the motif by removing flanking regions with less than 20% of the information content of the base with the highest information content in the motif. Positive and negative sequences are sampled randomly at training-time every time a batch is generated. We use simdna to generate simulated positive and negative sequences [13].

We train models with and without the Fourier-based prior over 3 random initializations each, and pick the model and epoch with the lowest validation loss for all downstream analyses.

2.9 Comparison to Basset

We compare the predictive performance of our single-task K562 binary models to Basset [14]. We download the model architecture and weights from Kipoi, a repository of predictive models for genomics [15]. We then use this restored Basset model to evaluate our K562 binary test set (Sec. 1.5), using all positive bins and an equally sized random sample of the negative bins. We compute the auROC and auPRC of the predictions for Basset, as well as our best-performing single-task K562 binary model (trained without any prior).

Note that while our binary models accept input sequences of 1000 bp, Basset takes in input sequences of 600 bp. When we test on Basset, we create our test set in the same way as described in Sec. 1.5, except we pad the sequence to length 600 bp rather than 1000 bp (i.e. the central 200 bp is still required to overlap a high-confidence peak by at least 100 bp to be considered a positive).

Additionally, Basset is trained on hg19, whereas our dataset is created using hg38 annotations. Because the inputs to the models are simply one-hot encoded sequences, however, it is expected that there is virtually no difference in the input data distributions.

3 DeepSHAP computation

To compute attributions (i.e. importance scores) for input sequences, we utilize DeepSHAP [16]. For binary models, we explain the binary prediction logits, summed across tasks. For profile models, we explain the profile prediction logits weighted by the final post-softmax probabilities, summed across the profile, and summed across tasks and strands.

Our baseline/reference set for DeepSHAP computation consists of 10 randomly shuffled versions of the input sequence, preserving dinucleotide frequencies. This choice of reference was recommended for genomic sequences in the DeepLIFT paper [17].

In subsequent sections, we use the term "hypothetical" DeepSHAP importance to refer to the estimated importance scores of an input sequence for all possible bases (i.e. the estimated DeepSHAP attributions for each base if it were present, hypothetically, at that location). The procedure for computing "hypothetical" importance scores can be found in Shrikumar et al. [18]. We use the term "actual" DeepSHAP importance to refer to the hypothetical importance multiplied by the input sequence (i.e. projected onto the bases that are actually in the sequence) and reduced to a single dimension along the input sequence by summing over the base identities at each position.

In order to make DeepSHAP work with PyTorch models and to easily produce "hypothetical" importance scores, we used a slightly modified version of the DeepSHAP library, available at `https://github.com/amtseng/shap`.

4 Signal-to-noise ratio of attributions

For each dataset, we select 1000 random positive examples from the test set and examine their interpretability. For profile models, we include a random jitter of ± 128 bp to avoid center-bias. We compute the "actual" DeepSHAP

importance scores as described above for each sequence and quantify each sequence's signal-to-noise ratio as the sum of the high-frequency normalized Fourier components, as well as the Shannon entropy. For simplicity, the sum of high-frequency normalized Fourier components is computed identically to the value of the Fourier-based prior loss (Sec. 2.3), but with $s = \infty$ (i.e. no softness in the frequency cut-off). The Shannon entropy is computed by taking the absolute value of the attributions and normalizing them along the sequence to pseudo-probabilities. We report the average high-frequency Fourier sum and the average Shannon entropy over the sampled test-set sequences, as well as the standard error of the mean.

We use this same procedure to compare the attributions of our SPI1 binary models trained with L2-regularization versus the Fourier-based prior.

5 Motif discovery and motif calling

For our Nanog/Oct4/Sox2 models, we call motifs using TF-MoDISco v0.5.5.5 [18]. On the entire set of IDR-thresholded peaks in the test set, we compute DeepSHAP importance scores on the individual tasks for each TF, and run TF-MoDISco. We utilize a sliding window size of 21, flank size of 10, and seqlet FDR of 0.01. From the resulting motif clusters, we select only the motifs that have an average information content of at least 0.6 over some window of length 6. We then trim the motifs by removing flanking regions with less than 20% of the information content of the base with the highest information content in the motif. We also remove motifs corresponding to homopolymer repeats. For binary models, due to their innate limitations in distinguishing primary motifs, we further filter motifs to have at least 750 supporting seqlets. For each motif, this gives us a PWM (Position Weight Matrix) and a CWM (Contribution Weight Matrix). The CWM consists of the aggregated "actual" DeepSHAP importance scores [3]. The background frequencies used to compute the PWM are 27% A or T, and 23% G or C.

To call motif instances, we select 1000 random positive examples from the test set. For profile models, we include a random jitter of ± 128 bp to avoid center-bias. For each motif, we compare it to every possible window in these sequences, and call a motif instance if:

1. The PWM match score is positive
2. The summed continuous Jaccard similarity (as defined below in Sec. 7) between the motif CWM and the underlying "actual" DeepSHAP scores of the sequence window is in the top 10% for that motif

This method of calling motif instances was adapted from Avsec et al. [3], which found that using CWMs to call motifs was significantly more effective for identifying high-confidence motifs (as supported by the experimental assay) compared to traditional PWM scanning.

We then rank the motif calls by the total DeepSHAP importance magnitude, summed across the instance, and compute precision–recall for which motif instances overlap with corresponding ChIP-nexus peaks by at least 1 base.

6 Performance on binary models

On our binary models, we compute accuracy, auROC, and auPRC on the test set, randomly subsampling negative bins to achieve balanced positives and negatives. We also compute an estimate of the auPRC on the imbalanced test set by artificially inflating the false positive rate to simulate auPRC computation on the entire test set. These are computed over all 30 random initializations for each dataset. The performance metrics of the best-performing model over the random initializations is recorded, with the best-performing model being the model with the lowest validation loss over all random initializations and epochs.

7 Stability of model learning

For each dataset, we select 100 random positive examples from the test set and examine the consistency of their "hypothetical" DeepSHAP importance across different models. For profile models, we include a random jitter of ± 128 bp to avoid center-bias.

For each input sequence, we compute the similarity of "hypothetical" DeepSHAP importance from any two models using the continuous Jaccard similarity metric [18]. The continuous Jaccard similarity is computed between the two sets of ℓ_1-normalized "hypothetical" attribution scores for each base, and the resulting similarities are summed across the length of the sequence. For two d-vectors u and v (here, $d = 4$ for the 4 bases), the Jaccard similarity per base is

computed as follows:

$$\sum_{i=1}^{d} \text{sign}(u_i)\text{sign}(v_i)\frac{\min\{|u_i|,|v_i|\}}{\max\{|u_i|,|v_i|\}}$$

To quantify the similarity across all 30 random initializations of a model, for each of the sampled input sequences, we compute the pairwise continuous Jaccard similarity sum for each pair of models and average the scores across all $\binom{30}{2}$ pairs. To quantify the similarity between models trained with all versus only 1% of the training data, we select the top 5 best-performing models when trained with the entire training set, and the top 5 best-performing models when trained with only 1% of the training set, and we compute the pairwise continuous Jaccard similarity sum across all 5×5 pairs between the two conditions. The best-performing models are selected by picking the 5 models (i.e. 5 different random initializations) in each condition with the best validation loss (or profile validation loss, for profile models). We report the average similarity over the sampled test-set sequences (where the similarity value for each sequence is the average continuous Jaccard score over many pairs of models), as well as the standard error of the mean.

8 Reliance on biologically relevant regions

For each dataset, we select 1000 random positive examples from the test set and examine the relationship between the "actual" DeepSHAP attributions and the underlying summits/peaks. For profile models, we include a random jitter of ± 128 bp to avoid center-bias.

First, over each of the 1000 sampled sequences, we compute the Spearman correlation of the magnitude of the "actual" DeepSHAP importance at each base to the distance to the closest IDR-thresholded peak summit. We report the average Spearman correlation over the sampled test-set sequences as well as the standard error of the mean.

Next, over all 1000 sampled sequences in aggregate, we rank all bases in descending order by the magnitude of "actual" DeepSHAP importance, and ask whether the top bases overlie IDR-thresholded peaks. We use this to compute precision–recall curves, where thresholds are the "actual" DeepSHAP magnitude, and a positive is when a base overlies an IDR-thresholded peak.

We use this same procedure as above to compare the attributions of our SPI1 binary models trained with L2-regularization versus the Fourier-based prior, and when comparing the Fourier-based prior with the smoothness and sparsity priors.

For the K562 models and Nanog/Oct4/Sox2 models, we perform further analyses using orthogonal footprints or ChIP-nexus data, respectively. The K562 footprints are computed by Vierstra et al. [19]. From their published data, we aggregated the footprints of all K562 experiments using BEDtools merge [6]. For the Nanog/Oct4/Sox2 models, in addition to ChIP-seq experiments, Avsec et al. [3] also performed ChIP-nexus experiments, and we utilize their IDR-thresholded peaks from the ChIP-nexus experiments.

To compute the fraction of importance in a ChIP-nexus peak or footprint, we take the absolute value of the "actual" DeepSHAP importance of each of the sampled sequences, and calculate the proportion (out of the total sum across that sequence) within a ChIP-nexus peak or DNase footprint. We also repeat the rank-based analysis with footprints or ChIP-nexus peaks instead of ChIP-seq or DNase-seq peaks. We report the average proportion over the sampled test-set sequences as well as the standard error of the mean.

For some of the binary models, we also compute the motif clusters of the high-importance regions that the Fourier-based prior highlighted, but where the regions did not overlie a ChIP-nexus peak or a footprint. Over the same sample of 1000 test sequences used to compute the auPRC of ChIP-nexus peak or footprint overlap, we take the top 200,000 most important bases (i.e. highest magnitude of DeepSHAP score) for which the base did not overlie a ChIP-nexus peak or a footprint. For each base, we expand to a centered 50 bp region, discarding overlaps (keeping the higher-ranked bases). For the resulting regions, we run the TF-MoDISco v0.5.5.5 [18] clustering algorithm only, clustering seqlets of length 30 into motifs.

To evaluate the ability of our models trained on simulated data to place importance only on motifs, we employ a similar procedure as above. We take a random sample of 100 simulated positive sequences (i.e. with placed motifs). For each sequence, we identify instances of the SPI1 motif by considering locations where the match score to the SPI1 PWM (i.e. the PWM used to construct the sequences) is over 0.9. The match score of a potential motif instance is computed as the sum of the entries in the PWM corresponding to the bases in the instance (i.e. total log-odds). Here, we use a uniform background (i.e. 25% A, C, T, or G) to compute the PWM. To compute the fraction of importance in the motifs, we use the same procedure as above to compute fraction of importance in a ChIP-nexus peak or footprint, but use motif instances instead of peaks/footprints. To compute the auPRC of base overlap with motif instances, we use the same

procedure as above to compute precision–recall of base overlap with peaks/footprints, but use called motif instances instead.

9 GC content simulations

We train single-task binary SPI1 models on varying amounts of GC bias. The procedure for constructing simulated sequences is identical to the procedure described above for training single-task binary SPI1 models without GC bias. The only difference is that when we synthesize a positive sequence (i.e. with an inserted motif instance), the background sequence can have a higher amount of GC content. The negative sequence set always has sequences of equal GC and AT content (i.e. no bias). We train models on 5 different levels of GC content: +0%, +1%, +2%, +3%, and +4%. A level of GC bias of +x% means the probability of G or C in the positive background sequences is $(50 + x)$%, while the negative sequence background has a G/C probability of 50% (i.e. no bias at all).

We train models with and without the Fourier-based prior over 3 random initializations each, and pick the model and epoch with the lowest validation loss for all downstream analyses.

To quantify how much a model is relying on GC content in the background, we sample 100 random positive sequences with the specified amount of GC bias, and compare the importance of A or T to the importance of G or C. For each sequence, we first mask out all instances of the SPI1 motif by ignoring any locations where the match score to the SPI1 PWM (i.e. total log-odds) is over 0.9. When scanning each simulated sequence for matches to the SPI1 motif, we compute the PWM using background frequencies that match the GC content of the sequence. We then consider the (signed) "hypothetical" DeepSHAP importance of A or T versus G or C, normalized by the maximum importance over the entire "hypothetical" importance track. We compute the product of A/T importance and G/C importance per base, and average over each input sequence.

References

[1] ENCODE Project Consortium et al. An integrated encyclopedia of DNA elements in the human genome. *Nature*, 489(7414): 57–74, Sep 2012. ISSN 14764687. doi: 10.1038/nature11247.

[2] Carrie A Davis, Benjamin C Hitz, Cricket A Sloan, Esther T Chan, Jean M Davidson, Idan Gabdank, Jason A Hilton, Kriti Jain, Ulugbek K Baymuradov, Aditi K Narayanan, Kathrina C Onate, Keenan Graham, Stuart R Miyasato, Timothy R Dreszer, J Seth Strattan, Otto Jolanki, Forrest Y Tanaka, and J Michael Cherry. The Encyclopedia of DNA elements (ENCODE): data portal update. *Nucleic Acids Research*, 46(D1):D794–D801, 2017. ISSN 0305-1048. doi: 10.1093/nar/gkx1081. URL `https://doi.org/10.1093/nar/gkx1081`.

[3] Žiga Avsec, Melanie Weilert, Avanti Shrikumar, Amr Alexandari, Sabrina Krueger, Khyati Dalal, Robin Fropf, Charles McAnany, Julien Gagneur, Anshul Kundaje, and Julia Zeitlinger. Deep learning at base-resolution reveals motif syntax of the cis-regulatory code. *bioRxiv*, page 737981, Aug 2019. doi: 10.1101/737981. URL `https://www.biorxiv.org/content/10.1101/737981v1`.

[4] Jinwook Lee, Daniel Kim, Grey Cristoforo, Chuan-Sheng Foo, Chris Probert, Nathan Beley, and Anshul Kundaje. ENCODE ATAC-seq pipeline. Dec 2019. doi: 10.5281/ZENODO.3564806.

[5] Galip Gürkan Yardimci, Christopher L. Frank, Gregory E. Crawford, and Uwe Ohler. Explicit DNase sequence bias modeling enables high-resolution transcription factor footprint detection. *Nucleic Acids Research*, 2014. ISSN 13624962. doi: 10.1093/nar/gku810.

[6] Aaron R. Quinlan and Ira M. Hall. BEDTools: A flexible suite of utilities for comparing genomic features. *Bioinformatics*, 2010. ISSN 13674803. doi: 10.1093/bioinformatics/btq033.

[7] Annashcherbina, Av Shrikumar, and Soumya Kundu. kundajelab/seqdataloader: v0.2. Apr 2020. doi: 10.5281/ZENODO.3771365.

[8] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, and Richard Durbin. The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, 2009. ISSN 13674803. doi: 10.1093/bioinformatics/btp352.

[9] W. J. Kent, A. S. Zweig, G. Barber, A. S. Hinrichs, and D. Karolchik. BigWig and BigBed: Enabling browsing of large distributed datasets. *Bioinformatics*, 2010. ISSN 13674803. doi: 10.1093/bioinformatics/btq351.

[10] Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not Just a Black Box: Learning Important Features Through Propagating Activation Differences. *arXiv*, 1:0–5, May 2016. URL `http://arxiv.org/abs/1605.01713`.

[11] Gabriel Erion, Joseph D. Janizek, Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Learning Explainable Models Using Attribution Priors. Jun 2019. URL `http://arxiv.org/abs/1906.10670`.

[12] Sven Heinz, Christopher Benner, Nathanael Spann, Eric Bertolino, Yin C. Lin, Peter Laslo, Jason X. Cheng, Cornelis Murre, Harinder Singh, and Christopher K. Glass. Simple Combinations of Lineage-Determining Transcription Factors Prime cis-Regulatory Elements Required for Macrophage and B Cell Identities. *Molecular Cell*, 2010. ISSN 10972765. doi: 10.1016/j.molcel.2010.05.004.

[13] Av Shrikumar, Johnny Israeli, Karl Leswing, Pgreenside, Bharath Ramsundar, Wainberg, and BenjaminrBond. kundajelab/simdna: Bugfix for simdata loading from non-gzipped files. Jun 2019. doi: 10.5281/ZENODO.3258813.

[14] 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, Jul 2016. ISSN 15495469. doi: 10.1101/gr.200535.115.

[15] Žiga Avsec, Roman Kreuzhuber, Johnny Israeli, Nancy Xu, Jun Cheng, Avanti Shrikumar, Abhimanyu Banerjee, Daniel Kim, Lara Urban, Anshul Kundaje, Oliver Stegle, and Julien Gagneur. Kipoi: accelerating the community exchange and reuse of predictive models for genomics. *bioRxiv*, page 375345, Jul 2018. doi: 10.1101/375345. URL `https://doi.org/10.1101/375345`.

[16] Scott M Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. In I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf`.

[17] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning Important Features Through Propagating Activation Differences, Jul 2017. ISSN 1938-7228. URL `http://proceedings.mlr.press/v70/shrikumar17a.html`.

[18] Avanti Shrikumar, Katherine Tian, Anna Shcherbina, Žiga Avsec, Abhimanyu Banerjee, Mahfuza Sharmin, Surag Nair, and Anshul Kundaje. TF-MoDISco v0.4.2.2-alpha: Technical Note. Oct 2018. URL `http://arxiv.org/abs/1811.00416`.

[19] Jeff Vierstra, John Lazar, Richard Sandstrom, Jessica Halow, Kristen Lee, Daniel Bates, Morgan Diegel, Douglas Dunn, Fidencio Neri, Eric Haugen, Eric Rynes, Alex Reynolds, Jemma Nelson, Audra Johnson, Mark Frerker, Michael Buckley, Rajinder Kaul, Wouter Meuleman, and John A. Stamatoyannopoulos. Global reference mapping and dynamics of human transcription factor footprints. *bioRxiv*, page 2020.01.31.927798, Feb 2020. doi: 10.1101/2020.01.31.927798.