Benchmarking Reverse-Complement Strategies for Deep Learning Models in Genomics

Hannah Zhou*
hannah.g.zhou@gmail.com

Avanti Shrikumar*†
avanti@cs.stanford.edu

Anshul Kundaje †
akundaje@stanford.edu

Abstract

Predictive models that map double-stranded regulatory DNA to molecular signals of regulatory activity should, in principle, produce identical predictions regardless of whether the sequence of the forward strand or its reverse complement (RC) is supplied as input. Unfortunately, standard convolutional neural network architectures can produce highly divergent predictions across strands, even when the training set is augmented with RC sequences. Two strategies have emerged in the literature to enforce this symmetry: conjoined a.k.a. "siamese" architectures where the model is run in parallel on both strands & predictions are combined, and RC parameter sharing or RCPS where weight sharing ensures that the response of the model is equivariant across strands. However, systematic benchmarks are lacking, and neither architecture has been adapted to base-resolution signal profile prediction tasks. In this work, we extend conjoined and RCPS models to signal profile prediction, and introduce a strong baseline: a standard model (trained on RC augmented data) that is converted to a conjoined model only after it has been trained, which we call a "post-hoc" conjoined model. We then conduct benchmarks on both binary and signal profile prediction. We find post-hoc conjoined models consistently perform as well as or better than models that were conjoined during training, and present a mathematical intuition for why. We also find that - despite its theoretical appeal - RCPS performs surprisingly poorly on certain tasks, in particular, signal profile prediction. In fact, RCPS can sometimes do worse than even standard models trained with RC data augmentation. We prove that the RCPS models can represent the solution learned by the conjoined models, implying that the poor performance of RCPS may be due to optimization difficulties. We therefore suggest that users interested in RC symmetry should default to post-hoc conjoined models as a reliable baseline before exploring RCPS.

1 Introduction

Convolutional neural networks (CNNs) have emerged as state-of-the-art models for predicting genome-wide regulatory signals such as transcription factor binding and chromatin accessibility as a function of DNA sequence [1–3]. The CNNs learn predictive motif syntax patterns encoded in the DNA sequences that orchestrate binding of transcription factor (TF) complexes. Unfortunately, the standard CNN architectures used for these predictions tasks do not explicitly account for the reverse complement equivalence of double-stranded regulatory DNA, where complementary base pairing implies that a pattern appearing on the forward strand is equivalent to one that appears in the reverse-complement (RC) orientation. These architectures can produce very different predictions depending on whether a strand is supplied in the forward vs. the RC orientation, even when the training dataset is augmented to contain reverse-complements [4]. Differing predictions erode confidence in model interpretation, as they could imply that motifs are missed on either the forward or reverse strands. Early work in deep learning for genomics enforced RC symmetry by combining model predictions across forward and RC versions of the input - for example, DeepBind [5] used the maximum prediction across both strands, while FactorNet [3] used the average. Such architectures

*co-first authors † co-corresponding authors.

are sometimes called conjoined a.k.a. "siamese" architectures [6]. This representation merging can be applied either during model training (as was done in [5, 3, 6]), or after the model has already been trained. Although it has not been described as such, the latter case is equivalent to converting a standard model to a conjoined one post-training (post-hoc). To date, no work has investigated whether trained conjoined models provide any benefits over post-hoc conjoined models.

A notable drawback of the conjoined architectures is that forward and RC versions of a non-palindromic motif must be learned as though they are two completely separate motifs - i.e. the model has no explicit knowledge of DNA double-strandedness. To address this, Shrikumar et al. [4] proposed RC parameter sharing (RCPS), where the weights of every convolutional filter are paired with those of another filter in such a way that each filter recognizes the reverse complement of what the other filter recognizes. Crucially, the filter at position c (0-indexed) in the channel axis is tied to the filter at position $C - 1 - c$, where C is the total number of filters in the layer. In doing so, RCPS extends the notion of a "reverse complement" to intermediate convolutional layers; when the RC operation consists of flipping a tensor both length-wise and channel-wise (which holds true when the DNA is one-hot encoded along the channel axis using the order ACGT), then under RCPS, applying the RC operation on an intermediate convolutional layer (i.e. flipping the length and channel axes) is equivalent to running the convolutional layer on the RC of the input sequence. Thus, RCPS layers are "equivariant" [7] under reverse-complementation. This idea of RCPS has been employed in several subsequent works: Brown and Lunter [8] extended RCPS to models with dropout and applied it to predict recombination hotspots, while Bartoszewicz et al. [6] applied it to predict the pathogenic potential of novel DNA. Nevertheless, to date, there has not been a systematic benchmark of RCPS against trained or post-hoc conjoined models that have similar architectures.

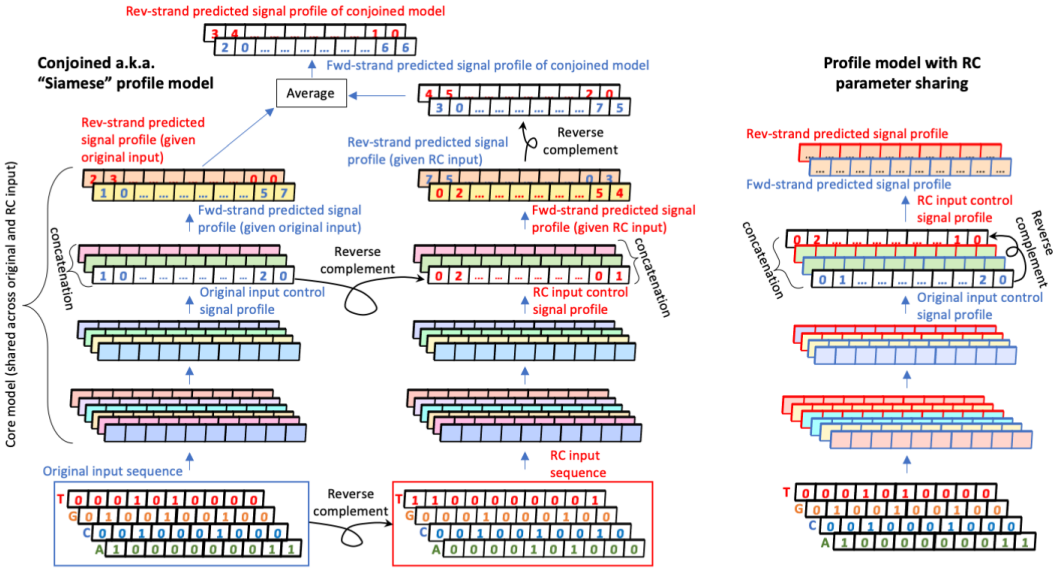

Figure 1: **Guaranteeing RC symmetry for the task of base-pair-level signal profile prediction**. Corresponding architectures for binary output models are in **Fig. S7**. In the standard BPNet profile prediction architecture, which is fully convolutional, the input control signal profile is concatenated as an extra channel to the activations of an intermediate convolutional layer, and separate predictions are made for positive and negative strands. **Left:** conjoined BPNet architecture for profile prediction. Cells with similar shading represent convolutional neurons with shared parameters. Unlike previously-proposed conjoined architectures for binary tasks, the model output on forward and RC inputs cannot simply be averaged; due to the strand-specific profile predictions, the output on the RC input needs to be reverse-complemented to be compatible with predictions on the forward input. **Right:** RCPS architecture for profile prediction. Cells with similar shading represent convolutional neurons with shared parameters; blue outlines denote the "forward" versions of a filter, while red outlines denote the corresponding RC versions. Unlike the RCPS architectures used in binary models, the forward and reverse filters are never collapsed into a single representation; this allows the model to make strand-specific predictions. To maintain RC equivariance, the input "control track" signal profile must be appended to both ends of the convolutional filter stack (once in the forward orientation and once in the RC orientation).

A third limitation of the existing literature on RC architectures is that they have not been extended to the task of base-pair level signal profile prediction, which has demonstrated immense potential to

learn high-resolution, higher-order *cis*-regulatory logic of TF binding [9]. The existing state-of-the-art method for profile prediction is the BPNet architecture [9], which predicts the shape of the observed signal (in the form of a probability distribution over a 1kbp interval) at base-pair resolution using both DNA sequence and a "control" (experimental bias) signal track as input. Separate predictions are made for the forward and reverse strands; this separation enables modeling of the asymmetric "strand shift" found at TF binding sites in profiles obtained from TF binding assays such as ChIP-seq and ChIP-nexus/exo. Extending the BPNet architecture to have RC symmetry would thus need to handle reverse complements at multiple stages of the input and output (**Fig. 1**).

In this work, we extend RC architectures to base-pair-level signal profile prediction and conduct systematic benchmarks across various tasks and datasets. We find that post-hoc conjoined models consistently perform as well as or better than trained conjoined models, and provide a mathematical intuition for why. We also prove that the representational capacity of the RCPS models encompasses that of a conjoined model, but nevertheless observe that the empirical performance of RCPS is poor relative to the post-hoc conjoined models, hinting at optimization difficulties. This work has important implications for practitioners looking to achieve RC symmetry in their models.

2 Methods

Full details on the architectures and datasets are provided in **Sec. S3**. In brief: we created two simulated datasets consisting of synthetic DNA sequences of lengths 200bp and 1kbp respectively, that contained motif instances sampled from 3 different TF motif models (Position Weight Matrices i.e. PWMs). Multi-task CNNs (with 3 binary output tasks) were trained to predict whether a given sequence contained instances of a particular motif. We also used genome-wide binarized TF-ChIP-seq data for Max, Ctcf and Spi1 in the GM12878 lymphoblastoid cell-line [4]. In these data, the positive set contained 1kbp sequences centered on high-confidence TF ChIP-seq peaks, and the negative set contained 1kbp sequences centered on chromatin accessible sites (DNase-seq peaks) in the same cell-lines that do not overlap any TF ChIP-seq peaks. Single-task binary output CNNs were trained on these data. For the base-pair-level signal profile prediction, we used genome-wide ChIP-nexus profiles of four TFs - Oct4, Sox2, Nanog and Klf4 in mouse embryonic stem cells [9]. BPNet-style models were trained with a multinomial loss to predict the distribution of reads in 1kbp regions for each of the two strands within ChIP-nexus peaks. Separate models were trained for each TF.

3 Results

3.1 Post-hoc conjoined models outperform trained conjoined models

Across all datasets, we found that post-hoc conjoined architectures (when trained with RC data augmentation) consistently perform as well as or better than trained conjoined architectures. In fact, post-hoc conjoined models achieved the best performance on the profile prediction tasks (**Fig. 2**). We present a mathematical intuition for why: consider a standard (non-conjoined) model f that gives a too-high prediction for an input sequence S and a too-low prediction on the reverse-complement S'. If the model were trained with data augmentation, both S and S' would be separate examples in the same batch, and the gradient descent update would raise $f(S')$ while lowering $f(S)$. By contrast, when training with the conjoined version of f, there would be no separate loss computed for $f(S)$ and $f(S')$; only $(f(S') + f(S))/2$ would be compared to the true value. Thus, if $(f(S') + f(S))/2$ were close to the true value, the gradient descent update for the conjoined model would not change $f(S')$ or $f(S)$, even though the model has not learned the true value in either case. Based on this reasoning, we recommend against using conjoined models during training.

3.2 RCPS has inconsistent performance across datasets

While we were able to replicate the strong performance of RCPS reported by Shrikumar et al. [4] on their binary classification datasets, we made a few interesting observations. First, RCPS on the 200bp simulated sequence dataset did not perform significantly better than the data-augmented post-hoc conjoined models (**Fig. S6**). Second, as discussed in **Sec. S2**, RCPS did not perform as well relative to other methods on the Max ChIP-seq task when the maximum training iterations was increased beyond what Shrikumar et al. [4] used. Third, on profile prediction models, RCPS *consistently* underperformed relative to post-hoc conjoined models, and did not significantly outperform data-

augmented standard models (**Fig. 2**). In fact, for the Oct4 profile prediction task, RCPS did not significantly outperform standard models trained *without* data augmentation. RCPS also had a high variance between random seeds on the Oct4 task.

These results are especially surprising given that the solution learned by the post-hoc conjoined models could have equivalently been represented using the RCPS models. The proof of this equivalence is given in **Sec. S1**. We thus conclude that the poor performance of RCPS is not due to a representational limitation, but due to RCPS failing to converge to a good solution. One hypothesis is that the RCPS models over-fit due to their greater representational capacity (this greater capacity comes from each filter having an RC version of itself in the convolutional layer - thus, the effective number of filters is twice that of a standard model). However, when we reduced the effective number of filters in the RCPS models to more closely resemble the number of filters in standard models, we consistently observed a drop in performance (**Sec. S3.4**), suggesting that over-fitting may not be the culprit.

4 Discussion

In this work, we showed that post-hoc conjoined models with RC data augmentation reliably performs on par or better than their trained conjoined counterparts. Overall, post-hoc conjoined models achieved the best or second-best performance across all datasets, surpassed only by RCPS on select datasets. Unfortunately, RCPS was unreliable, in that it frequently failed to outperform standard models trained with RC data augmentation - particularly for profile prediction. Given that the RCPS models are capable of representing the solution learned by the conjoined models, we hypothesize that their poor performance is due to optimization difficulties. As qualitative support, we note that Brown and Lunter [8] used a non-standard initialization of the final output layer to encourage their RCPS models to converge to good solutions. Even so, Brown and Lunter [8] found RCPS did not significantly outperform standard models trained with RC data augmentation on *in vivo* binding data, even though RCPS did better on simulated data. These apparent difficulties may prevent the full potential of RCPS from being realized out-of-the-box. We thus recommend that deep learning practitioners exercise caution when adopting RCPS into their architectures, and always make sure to compare RCPS against a baseline of post-hoc conjoined models.

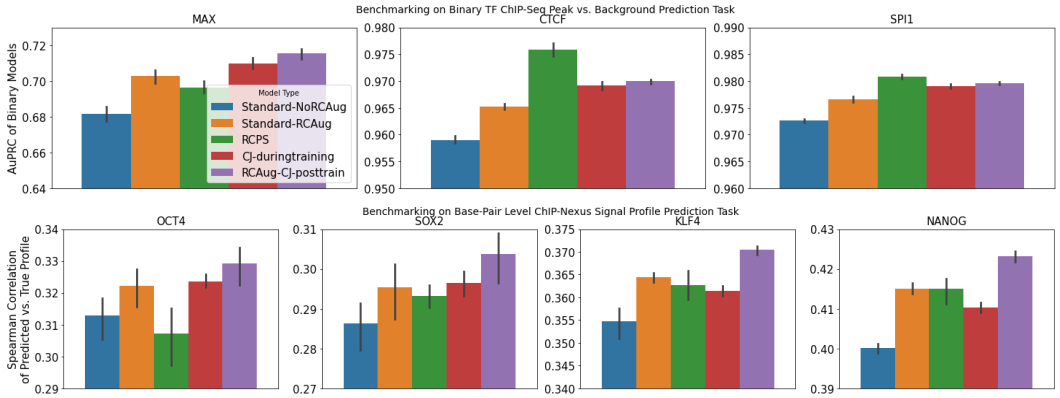

Figure 2: **Benchmarking models on TF ChIP-seq binary peak prediction and base-pair-level signal profile prediction tasks**. Bar heights represent the average performance over 10 random seeds, and error bars represent the 95% confidence intervals for the mean generated by seaborn [10] using 1000 bootstrapped samples. "Standard-RCAug" and "Standard-noRCAug" are standard models trained with and without RC data augmentation. RCPS denotes RC parameter sharing. CJ-duringtraining are models that were conjoined during training. RCAug-CJ-posttrain are standard models trained with RC data augmentation that were converted to conjoined models after training (i.e. post-hoc conjoined). RCAug-CJ-posthoc consistently performs as well or better than all other methods on all tasks except CTCF and SPI1 (where RCPS does best). However, RCPS performs no better than StandardRCAug on all profile prediction tasks and the Max binary task, and no better than Standard-noRCAug on the Oct4 profile prediction task. For the binary TF ChIP-seq models, the training hyperparameters were set to the tested combination that gave the best overall AuPRCs: 8000 (rather than 4000) max training iterations, AuROC (rather than model loss) as the metric used to choose the best validation-set epoch, and upsampling (rather than upweighting) of positives examples to achieve a 1:4 class ratio (see **Sec. S2** for more details). The corresponding plots for the simulated data are in **Fig. S6**. Plots showing different performance metrics for profile prediction are in **Fig. S9** , and the profile metrics are explained in **Sec. S3.2.3**.

5 Acknowledgments

We thank Alex M. Tseng for code to compute the performance metrics of profile prediction models. We thank Kelly Cochran for feedback on the manuscript.

References

[1] Jian Zhou and Olga G Troyanskaya. Predicting effects of noncoding variants with deep learning–based sequence model. *Nat. Methods*, 12(10):931–934, August 2015.

[2] David R Kelley, Jasper Snoek, and John Rinn. Basset: Learning the regulatory code of the accessible genome with deep convolutional neural networks. Technical report, October 2015.

[3] Daniel Quang and Xiaohui Xie. FactorNet: A deep learning framework for predicting cell type specific transcription factor binding from nucleotide-resolution sequential data. *Methods*, 166: 40–47, August 2019.

[4] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Reverse-complement parameter sharing improves deep learning models for genomics. *bioRxiv*, page 103663, January 2017.

[5] Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nat. Biotechnol.*, 33(8):831–838, August 2015.

[6] Jakub M Bartoszewicz, Anja Seidel, Robert Rentzsch, and Bernhard Y Renard. DeePaC: predicting pathogenic potential of novel DNA with reverse-complement neural networks. *Bioinformatics*, 36(1):81–89, January 2020.

[7] Taco Cohen and Max Welling. Group equivariant convolutional networks. pages 2990–2999. PMLR, June 2016.

[8] Richard C Brown and Gerton Lunter. An equivariant bayesian convolutional network predicts recombination hotspots and accurately resolves binding motifs. *Bioinformatics*, 35(13):2177–2184, July 2019.

[9] Žiga Avsec, Melanie Weilert, Avanti Shrikumar, Sabrina Krueger, Amr Alexandari, Khyati Dalal, Robin Fropf, Charles McAnany, Julien Gagneur, Anshul Kundaje, and Julia Zeitlinger. Base-resolution models of transcription factor binding reveal soft motif syntax. July 2020.

[10] Michael Waskom, Olga Botvinnik, Drew O'Kane, Paul Hobson, Saulius Lukauskas, David C Gemperline, Tom Augspurger, Yaroslav Halchenko, John B. Cole, Jordi Warmenhoven, Julian de Ruiter, Cameron Pye, Stephan Hoyer, Jake Vanderplas, Santi Villalba, Gero Kunter, Eric Quintero, Pete Bachant, Marcel Martin, Kyle Meyer, Alistair Miles, Yoav Ram, Tal Yarkoni, Mike Lee Williams, Constantine Evans, Clark Fitzgerald, Brian, Chris Fonnesbeck, Antony Lee, and Adel Qalieh. mwaskom/seaborn: v0.8.1 (september 2017), September 2017. URL `https://doi.org/10.5281/zenodo.883859`.

[11] Avant Shrikumar, Nic Fishman, and Anshul Kundaje. kundajelab/simdna: simulated datasets of DNA, June 2019. URL `https://doi.org/10.5281/zenodo.3258813`.

[12] Pouya Kheradpour and Manolis Kellis. Systematic discovery and characterization of regulatory motifs in ENCODE TF binding experiments. *Nucleic Acids Res.*, 42(5):2976–2987, March 2014.

[13] ENCODE Project Consortium. An integrated encyclopedia of DNA elements in the human genome. *Nature*, 489(7414):57–74, September 2012.

[14] 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. *Genome Res.*, 22(9):1813–1831, September 2012.

[15] Roadmap Epigenomics Consortium, Anshul Kundaje, Wouter Meuleman, Jason Ernst, Misha Bilenky, Angela Yen, Alireza Heravi-Moussavi, Pouya Kheradpour, Zhizhuo Zhang, Jianrong Wang, Michael J Ziller, Viren Amin, John W Whitaker, Matthew D Schultz, Lucas D Ward, Abhishek Sarkar, Gerald Quon, Richard S Sandstrom, Matthew L Eaton, Yi-Chieh Wu, Andreas R Pfenning, Xinchen Wang, Melina Claussnitzer, Yaping Liu, Cristian Coarfa, R Alan Harris, Noam Shoresh, Charles B Epstein, Elizabeta Gjoneska, Danny Leung, Wei Xie, R David Hawkins, Ryan Lister, Chibo Hong, Philippe Gascard, Andrew J Mungall, Richard Moore, Eric Chuah, Angela Tam, Theresa K Canfield, R Scott Hansen, Rajinder Kaul, Peter J Sabo, Mukul S Bansal, Annaick Carles, Jesse R Dixon, Kai-How Farh, Soheil Feizi, Rosa Karlic, Ah-Ram Kim, Ashwinikumar Kulkarni, Daofeng Li, Rebecca Lowdon, Ginell Elliott, Tim R Mercer, Shane J Neph, Vitor Onuchic, Paz Polak, Nisha Rajagopal, Pradipta Ray, Richard C Sallari, Kyle T Siebenthall, Nicholas A Sinnott-Armstrong, Michael Stevens, Robert E Thurman, Jie Wu, Bo Zhang, Xin Zhou, Arthur E Beaudet, Laurie A Boyer, Philip L De Jager, Peggy J Farnham, Susan J Fisher, David Haussler, Steven J M Jones, Wei Li, Marco A Marra, Michael T McManus, Shamil Sunyaev, James A Thomson, Thea D Tlsty, Li-Huei Tsai, Wei Wang, Robert A Waterland, Michael Q Zhang, Lisa H Chadwick, Bradley E Bernstein, Joseph F Costello, Joseph R Ecker, Martin Hirst, Alexander Meissner, Aleksandar Milosavljevic, Bing Ren, John A Stamatoyannopoulos, Ting Wang, and Manolis Kellis. Integrative analysis of 111 reference human epigenomes. *Nature*, 518(7539):317–330, February 2015.

[16] Tommy W Terooatea, Amir Pozner, and Bethany A Buck-Koehntop. PAtCh-Cap: input strategy for improving analysis of ChIP-exo data sets and beyond. *Nucleic Acids Res.*, 44(21):e159, December 2016.

S1 RCPS models can be used to represent equivalent conjoined models

In this section, we will show that the RCPS models considered in this work are capable of representing the solutions learned by the corresponding conjoined models that they are benchmarked against. The high-level intuition for our approach is as follows: recall that for each filter in an RCPS model, a corresponding "reverse-complement" filter is created through weight sharing. We will design the weights of our RCPS model such that (1) the activations of the RCPS "forward" convolutional filters on an input S match up with the activations of the standard model's convolutional filters on input S, and (2) the activations of the RCPS "reverse-complement" convolutional filters on an input S match up with the activations of the standard model's convolutional filters on S' (where S' is the reverse-complement of S). We will then show how there is a mapping between linear operations in the RCPS model (between the convolutional layers and the final output) and the step where the conjoined model averages the output of the standard model on S and S'. Thus, the RCPS model can represent any function learned by the corresponding conjoined model.

First, we will recap how RCPS models are constructed. Let $\mathbf{W}^{l,c}$ denote the weights of convolutional channel c (0-indexed) in layer l of the RCPS model, and let C_l denote the total number of channels (including additional reverse-complement channels generated by RCPS) in layer l of the RCPS model. The matrix $\mathbf{W}^{l,c}$ has dimensions of (w, C_{l-1}), where we use w to denote the width of the convolutional filters (without loss of generality, we will assume all layers use filters of the same width w; we will also set $C_0 = 4$ to represent the number of channels used in the one-hot encoding of ACGT in the input layer). Under RCPS, the weights of $\mathbf{W}^{l,c}$ are tied to the weights of $\mathbf{W}^{l,C_l-1-c}$. Specifically, if we used $W^{l,c}_{i,j}$ to denote the convolutional kernel weight on position i and input channel j, and use $b^{l,c}$ to denote the bias term for channel c in layer l, then we have:

$$\begin{aligned} W^{l,c}_{i,j} &= W^{l,C_l-1-c}_{w-1-i,C_{l-1}-1-j} \\ b^{l,c} &= b^{l,C_l-1-c} \end{aligned} \tag{1}$$

This weight sharing ensures that filter $(l, C_l - 1 - c)$ will recognize the reverse-complement of whichever pattern is recognized by filter (l, c).

Now, let us extend this notation to the corresponding standard models. Let $\mathbf{W}^{*,l,c}$ denote the weights of convolutional channel c at layer l of the standard model, and let C^*_l denote the total number of channels in layer l of the standard model. In all our benchmarks, we had $C_l = 2C^*_l$ (this was due to the duplication of filters in the RCPS models caused by reverse-complement weight sharing; we also ran comparisons where $C_l = C^*_l$ (**Sec. S3.4**) - however, when $C_l = C^*_l$, the equivalence explained in this section does not hold). Let us further use $A^l_{i,j}(S)$ and $A^{*,l}_{i,j}(S)$ to denote the activations in layer l, position i, channel j for the RCPS and standard model respectively when sequence S is supplied as input. When $l = 0$ (denoting the input layer), we will pretend $A^0_{i,j}$ and $A^{*,0}$ are simply the identity function. Let us also use S' to denote the reverse-complement of S, and let L_l denote the length of layer l.

If we set the weights for channels $c = 0$ through $c < C^*_l$ in the convolutional layers of the RCPS model such that:

$$\begin{aligned} W^{l,c}_{i,j} &= \begin{cases} W^{*,l,c}_{i,j} & j < C^*_{l-1}, c < C^*_l \\ 0 & j \geq C^*_{l-1}, c < C^*_l \end{cases} \\ b^{l,c} &= \begin{cases} b^{*,l,c} & c < C^*_l \end{cases} \end{aligned} \tag{2}$$

And if we also set the weights for channels $c \geq C^*_l$ through $c < C_l$ in accordance with RCPS weight sharing (**Eqn. 1**) such that:

$$\begin{aligned} W^{l,c}_{i,j} &= \begin{cases} 0 & j < C^*_{l-1}, c \geq C^*_l \\ W^{*,l,C_l-1-c}_{w-1-i,C_{l-1}-1-j} & j \geq C^*_{l-1}, c \geq C^*_l \end{cases} \\ b^{l,c} &= \begin{cases} b^{*,l,C_l-1-c} & c \geq C^*_l \end{cases} \end{aligned} \tag{3}$$

Then we can prove that:

$$A^l_{i,j}(S) = \begin{cases} A^{*,l}_{i,j}(S) & j < C^*_l \\ A^{*,l}_{L_l-1-i,C_l-1-j}(S') & j \geq C^*_l \end{cases} \tag{4}$$

S1.1 Proof of Eqn. 4

We will prove this by induction. We note that **Eqn. 4** holds true in the case of the input layer $l = 0$, assuming that one-hot encoding was done using the ordering ACGT. As mentioned, A^0 and $A^{*,0}$ are simply the identity function, so $A^0_{i,j}(S) = A^{*,l}_{i,j}(S) = S_{i,j}$. **Eqn. 4** simply states that if there is an A ($j = 0$) or a C ($j = 1$) at position i in the input sequence S, there will respectively be a T ($C_l - j - 1 = 4 - 1 = 3$) or a G ($C_l - j - 1 = 4 - 2 = 2$) at position $L_0 - 1 - i$ of the reverse-complement S'. Here L_0 is simply the length of the input sequence. Thus, **Eqn. 4** holds for the base-case of $l = 0$ due to the reverse-complement property of DNA.

We will now show that if **Eqn. 4** holds for the base-case of $l = 0$, it holds for all $l > 0$. From the definition of a convolutional operation, we have:

$$A^l_{i,j} = \sigma\left(b^{l,j} + \sum_{k=0}^{k<w}\sum_{c=0}^{c<C_{l-1}} W^{l,j}_{k,c} A^{l-1}_{i+k,c}\right) \tag{5}$$

Where σ denotes a nonlinearity. Let us begin by proving the case where j (the index of the convolutional channel) satisfies $j < C^*_l$ (i.e. j is in the first half of the convolutional filters - recall that $C_l = 2C^*_l$). From **Eqn. 2**, we have:

$$\begin{aligned}
A^l_{i,j}(S) &= \sigma\left(b^{*,l,j} + \sum_{k=0}^{k<w}\left(\sum_{c=0}^{c<C^*_{l-1}} W^{*,l,j}_{k,c} A^{l-1}_{i+k,c}(S) + \sum_{c=C^*_{l-1}}^{c<C_{l-1}} 0 \times A^{l-1}_{i+k,c}(S)\right)\right) \\
&= \sigma\left(b^{*,l,j} + \sum_{k=0}^{k<w}\sum_{c=0}^{c<C^*_{l-1}} W^{*,l,j}_{k,c} A^{l-1}_{i+k,c}(S)\right) \\
&= \sigma\left(b^{*,l,j} + \sum_{k=0}^{k<w}\sum_{c=0}^{c<C^*_{l-1}} W^{l,j}_{k,c} A^{*,l-1}_{i+k,c}(S)\right) \text{ (From \textbf{Eqn. 4}, by induction)} \\
&= A^{*,l}_{i,j}(S) \text{ (From the definition of a convolution)}
\end{aligned}$$

Let us now prove the case where $j \geq C^*_l$. Substituting **Eqn. 3**, we have:

$$\begin{aligned}
A^l_{i,j}(S) &= \sigma\left(b^{*,l,j} + \sum_{k=0}^{k<w}\left(\sum_{c=0}^{c<C^*_{l-1}} 0 \times A^{l-1}_{i+k,c}(S) + \sum_{c=C^*_{l-1}}^{c<C_{l-1}} W^{*,l,C_l-1-j}_{w-1-k,C_{l-1}-1-c} A^{l-1}_{i+k,c}(S)\right)\right) \\
&= \sigma\left(b^{*,l,j} + \sum_{k=0}^{k<w}\sum_{c=C^*_{l-1}}^{c<C_{l-1}} W^{*,l,C_l-1-j}_{w-1-k,C_{l-1}-1-c} A^{l-1}_{i+k,c}(S)\right) \\
&= \sigma\left(b^{*,l,C_l-1-j} + \sum_{k=0}^{k<w}\sum_{c=C^*_{l-1}}^{c<C_{l-1}} W^{*,l,C_l-1-j}_{w-1-k,C_{l-1}-1-c} A^{*,l-1}_{L_{l-1}-1-(i+k),C_{l-1}-1-c}(S')\right) \text{ (From \textbf{Eqn. 4}, by induction)} \\
&= A^{*,l}_{L_l-1-i,C_l-1-j}(S') \text{ (From the definition of a convolution)}
\end{aligned}$$

Thus, we have proven **Eqn. 4** for any layer in a stack of convolutions, extending from the input upwards. In words, this shows that the convolutional activations of the standard model on the forward

sequence S correspond to the convolutional activations of the "forward" filters of the RCPS model on the sequence S, and that the convolutional activations of the standard model on the RC sequence S' correspond to the convolutional activations of the "RC" filters of the RCPS model on sequence S.

S1.2 Combining the representations on the forward and reverse strands

Let us now consider how the conjoined model combines the representations on the forward and reverse strands. In binary conjoined models, the stack of convolutional layers is followed by a linear transformation that predicts the logit of the sigmoid, after which the representations from both strands are averaged and passed through the sigmoid. Specifically, if we use $\hat{l}$ to denote the last convolutional layer, g to denote the function computing the logit in the standard model, and we use $\mathbf{W}^{*,\mathbf{g}}$ & $b^{*,g}$ to denote the weights & biases of g, we have:

$$g^*(S) = b^{*,g} + \sum_{i,j} W^{*,g}_{i,j} A^{*,\hat{l}}_{i,j}(S)$$

Thus, the corresponding output $g^{**}(S)$ of the conjoined model is:

$$\begin{aligned} g^{**}(S) &= 0.5 \left(\left(b^{*,g} + \sum_i \sum_{j=0}^{j<C^*_{\hat{l}}} W^{*,g}_{i,j} A^{*,\hat{l}}_{i,j}(S) \right) + \left(b^{*,g} + \sum_i \sum_{j=0}^{j<C^*_{\hat{l}}} W^{*,g}_{i,j} A^{*,\hat{l}}_{i,j}(S') \right) \right) \\ &= b^{*,g} + 0.5 \left(\sum_i \sum_{j=0}^{j<C^*_{\hat{l}}} W^{*,g}_{i,j} \left(A^{*,\hat{l}}_{i,j}(S) + A^{*,\hat{l}}_{i,j}(S') \right) \right) \end{aligned}$$

In the case of the RCPS binary models, the "forward" and "RC" channels at the end of the stack of convolutional layers are added together (after reverse-complementing the RC channels to be compatible with the forward channels - see **Fig. S7**), and then a linear operation is applied to obtain the logit of the sigmoid. If we let g denote the function computing the logit of the RCPS model, and use $\mathbf{W}^{\mathbf{g}}$ & b^g to denote the weights and biases of g, we have:

$$\begin{aligned} g(S) &= b^g + \sum_i \sum_{j=0}^{j<\frac{C_{\hat{l}}}{2}} W^g_{i,j} \left(A^{\hat{l}}_{i,j}(S) + A^{\hat{l}}_{L_{\hat{l}}-1-i,C_{\hat{l}}-1-j}(S) \right) \\ &= b^g + \sum_i \sum_{j=0}^{j<\frac{C_{\hat{l}}}{2}} W^g_{i,j} \left(A^{*,\hat{l}}_{i,j}(S) + A^{*,\hat{l}}_{i,j}(S') \right) \text{ (From } \textbf{Eqn. 4}\text{)} \end{aligned}$$

The reason we iterate from $j = 0$ to $j < \frac{C_{\hat{l}}}{2}$ is that the effective number of filters in the RCPS model gets halved when the forward and reverse-complement channels are added together. Also recall that $\frac{C_{\hat{l}}}{2} = C^*_{\hat{l}}$. If we therefore set $b^g = b^{*,g}$ and $W^g_{i,j} = 0.5 W^{*,g}_{i,j}$, we will achieve $g(S) = g^{**}(S)$.

In the case of the profile prediction models, there is an additional nuance that separate predictions are made for the two output strands. For simplicity, we will treat the control bias track as though it is another channel at the end of the last nonlinear convolutional layer. We will denote the activations of the last nonlinear convolutional layer in the standard profile prediction model using $A^{*,\hat{l}}$. Since the operations following the last nonlinear convolutional layer are all linear convolutions (see **Sec. S3.2.2**), we will represent the output of the strands at position i for the standard model as:

$$A^*_{i,+}(S) = b^{*,+} + \sum_{k=0}^{k<w} \sum_{c=0}^{c<C^*_{\hat{l}}} W^{*,+}_{k,c} A^{*,\hat{l}}_{i+k,c}(S)$$

$$A^*_{i,-}(S) = b^{*,-} + \sum_{k=0}^{k<w} \sum_{c=0}^{c<C^*_{\hat{l}}} W^{*,-}_{k,c} A^{*,\hat{l}}_{i+k,c}(S)$$

After the reverse strand predictions are flipped and averaged with the forward strand, the output of the strands for the conjoined model is:

$$A^{**}_{i,+} = 0.5\left(b^{*,+} + b^{*,-} + \sum_{k=0}^{k<w}\sum_{c=0}^{c<C^*_{\hat{l}}}\left(W^{*,+}_{k,c}A^{*,\hat{l}}_{i+k,c}(S) + W^{*,-}_{w-1-k,c}A^{*,\hat{l}}_{L_{\hat{l}}-1-(i+k),c}(S')\right)\right)$$

$$A^{**}_{i,-} = 0.5\left(b^{*,+} + b^{*,-} + \sum_{k=0}^{k<w}\sum_{c=0}^{c<C^*_{\hat{l}}}\left(W^{*,+}_{w-1-k,c}A^{*,\hat{l}}_{L_{\hat{l}}-1-(i+k),c}(S') + W^{*,-}_{k,c}A^{*,\hat{l}}_{i+k,c}(S)\right)\right)$$

By comparison, the output of the forward strand for the RCPS model is written as:

$$\begin{aligned}A_{i,+} &= b^{+} + \sum_{k=0}^{w}\sum_{c=0}^{c<C_{\hat{l}}} W^{+}_{k,c}A^{\hat{l}}_{i+k,c}(S) \\ &= b^{+} + \sum_{k=0}^{w}\sum_{c=0}^{c<C^*_{\hat{l}}} W^{+}_{k,c}A^{*,\hat{l}}_{i+k,c}(S) + \sum_{c=C^*_{\hat{l}}}^{c<C_{\hat{l}}} W^{+}_{k,c}A^{*,\hat{l}}_{L_{\hat{l}}-1-(i+k),C_{\hat{l}-1-c}}(S') \text{ (By } \textbf{Eqn. 4}\text{)}\end{aligned}$$

If we thus set the weights such that $b^{+} = 0.5(b^{*,+} + b^{*,-})$ and

$$W^{+}_{k,c} = \begin{cases} 0.5W^{*,+}_{k,c} & c < C^*_{\hat{l}} \\ 0.5W^{*,-}_{w-1-k,C_l-1-c} & c \geq C^*_{\hat{l}} \end{cases}$$

We achieve $A_{i,+} = A^{**}_{i,+}$. The weights for $A_{i,-}$ would be given by **Eqn. 2**, and a similar calculation can be done to show $A_{i,-} = A^{**}_{i,-}$. Thus, the RCPS models can be used to represent the functions learned by the corresponding conjoined models.

S2 Some datasets show sensitivity to training hyperparameters

Previous analyses Shrikumar et al. [4] had shown that RCPS architectures outperformed standard architectures with RC data augmentation on several binary output TF binding prediction tasks. However, we found that increasing the maximum number of training iterations (i.e. batch updates) from 4000 (used by Shrikumar et al. [4]) to 8000, resulted in the standard architectures with RC data augmentation outperforming the RCPS architecture for predicting binary binding of the Max protein (**Fig. S3**). This trend was not observed for Ctcf (**Fig. S4**) and Spi1 proteins (**Fig. S5**), where RCPS remained in the lead irrespective of the limit on the number of training iterations. The Max dataset differs most dramatically from Spi1 and Ctcf in that it is has a more acute class imbalance (Max has 16.47:1 positive to negative example ratio compared to 4.75:1 for Spi1 and 5.01:1 for Ctcf). We hypothesize that due to the smaller number of positive training examples, the Max task is inherently harder to learn on, which may explain why the standard architecture needs more training iterations to find a good solution.

Inspired by this result, we explored the effect of other hyperparameters that impact model training (**Fig. S3, S5, S4**). For example, we explored the impact of the metric used to select the "best" validation set epoch. By default, the Keras "early stopping" callback selects the epoch with the best validation set loss. We found that this default approach could occasionally yield significantly lower validation-set auROCs and auPRCs relative to using the epoch with the best validation-set auROC, particularly for the Max dataset. Note that the approach of selecting the best validation set epoch using auROC was used in Shrikumar et al. [4], and we found it was necessary to replicate their results. We also found that when the training-set class imbalance was handled by up**sampling** positive examples to achieve a positives:negatives ratio of 1:4 (instead of up**weighting** positive examples in the loss function according to the class ratios, as was done in Shrikumar et al. [4]), the model auPRCs generally improved for the all three tasks. However, regardless of these hyerparameter choices, the main trends described in this work were robust to these differences in hyperparameters.

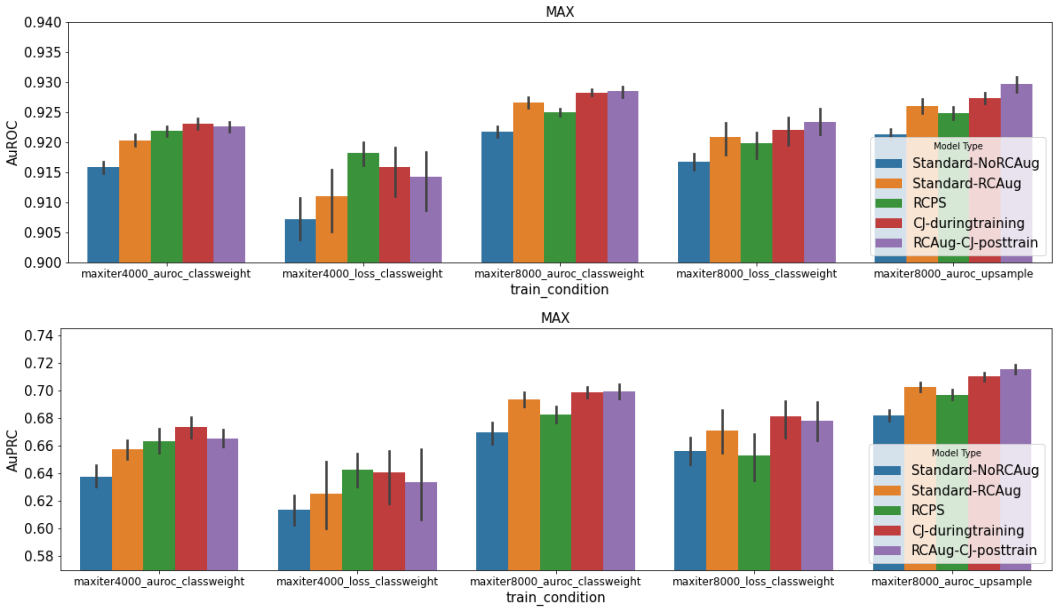

Figure S3: **Benchmarking effect of different training hyperparameters for Max TF ChIP-seq binary peak prediction.** Bar heights represent the average performance over 10 random seeds, and error bars represent the 95% confidence intervals for the mean generated by seaborn [10] using 1000 bootstrapped samples. "Standard-RCAug" and "Standard-noRCAug" are standard models trained with and without RC data augmentation. RCPS denotes RC parameter sharing. CJ-duringtraining are models that were conjoined during training. RCAug-CJ-posttrain are standard models trained with RC data augmentation that were converted to conjoined models after training (i.e. post-hoc conjoined).

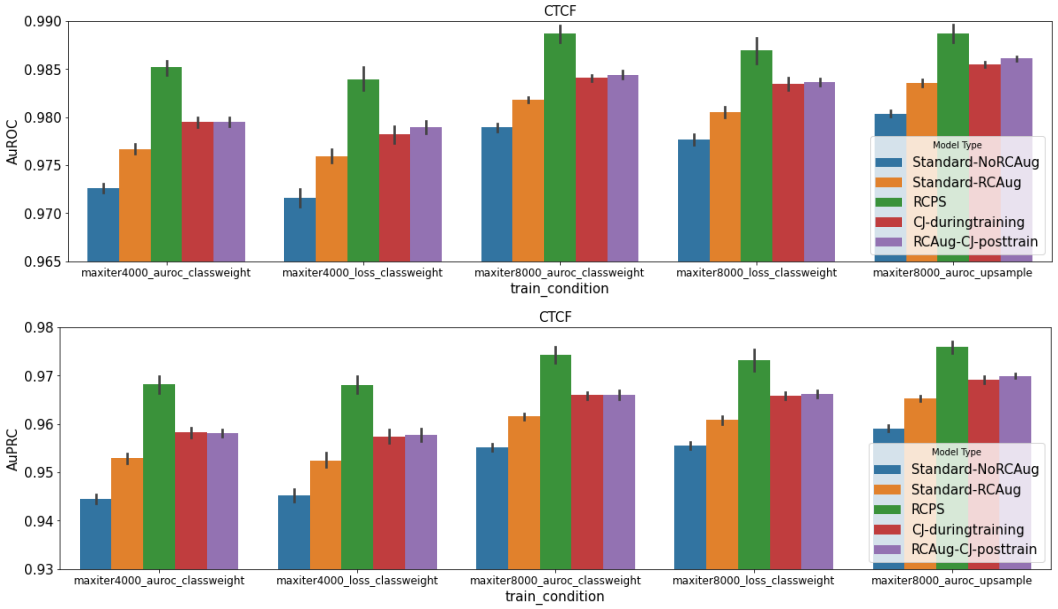

Figure S4: **Benchmarking effect of different training hyperparameters for Ctcf TF ChIP-seq binary peak prediction** Bar heights represent the average performance over 10 random seeds, and error bars represent the 95% confidence intervals for the mean generated by seaborn [10] using 1000 bootstrapped samples. "Standard-RCAug" and "Standard-noRCAug" are standard models trained with and without RC data augmentation. RCPS denotes RC parameter sharing. CJ-duringtraining are models that were conjoined during training. RCAug-CJ-posttrain are standard models trained with RC data augmentation that were converted to conjoined models after training (i.e. post-hoc conjoined).

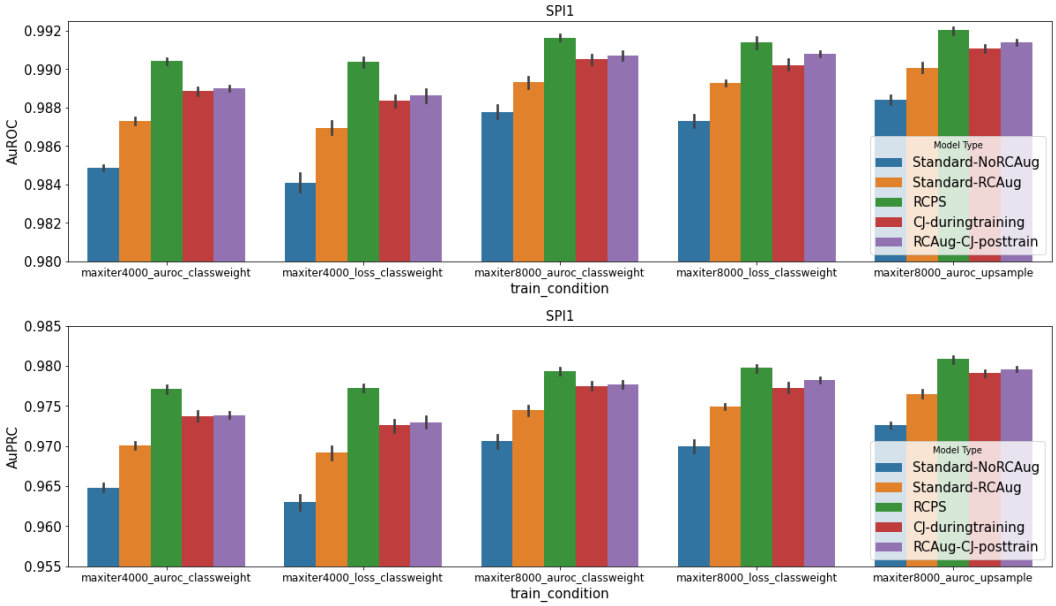

Figure S5: **Benchmarking effect of different training hyperparameters for Spi1 TF ChIP-seq binary peak prediction.** Bar heights represent the average performance over 10 random seeds, and error bars represent the 95% confidence intervals for the mean generated by seaborn [10] using 1000 bootstrapped samples. "Standard-RCAug" and "Standard-noRCAug" are standard models trained with and without RC data augmentation. RCPS denotes RC parameter sharing. CJ-duringtraining are models that were conjoined during training. RCAug-CJ-posttrain are standard models trained with RC data augmentation that were converted to conjoined models after training (i.e. post-hoc conjoined).

S3 Datasets & Models

S3.1 Binary prediction

S3.1.1 Binary prediction datasets

Our simulated datasets were based on those used in Shrikumar et al. [4], but with a few slight modifications. Three "sets" of sequences were generated using the simdna package [11]. One set contained instances of the GATA_known6 Position Weight Matrix (PWM), one set contained instances of the ELF1_known2 PWM, and one set contained instances of RXRA_known1 (the motifs were taken from the ENCODE consortium [12]). 10,000 sequences were generated for each set. The sequences were simulated as follows: (1) generate a random background sequence of length Xbp (where X was either 200 or 100) with 40% GC content (2) determine the number of instances of the motif to insert by sampling from a truncated Poisson distribution with mean 2, max 3 and min 1 (3) sample the motif instances from the specified PWM (either GATA_known6, RXRA_known1 or ELF1_known1), (4) reverse complement each sampled instance with probability 0.5 (5) insert each sampled instance at a random position within the sequence with the constraint that it does not overlap a previously inserted instance. We randomly allocated 30% of the data to validation, 30% to testing, and 40% to training. Labels where then generated as follows: there were three binary tasks, each task corresponding to a particular set; a sequence was labeled as a 1 for a task if it originated from the corresponding set, and with a 0 otherwise (note that while such a labeling scheme could be modeled using a softmax output and a categorical cross-entropy loss, we trained our models with three sigmoid outputs and used a binary cross-entropy loss for consistency with common deep learning architectures used in the literature). Finally, we simulated mislabeling noise by flipping each individual label with 20% probability (this approach of adding noise differed slightly from the approach used in [4] in that, in our case, the probability of flipping the label for a particular task was independent of the probability of flipping the label for the other tasks).

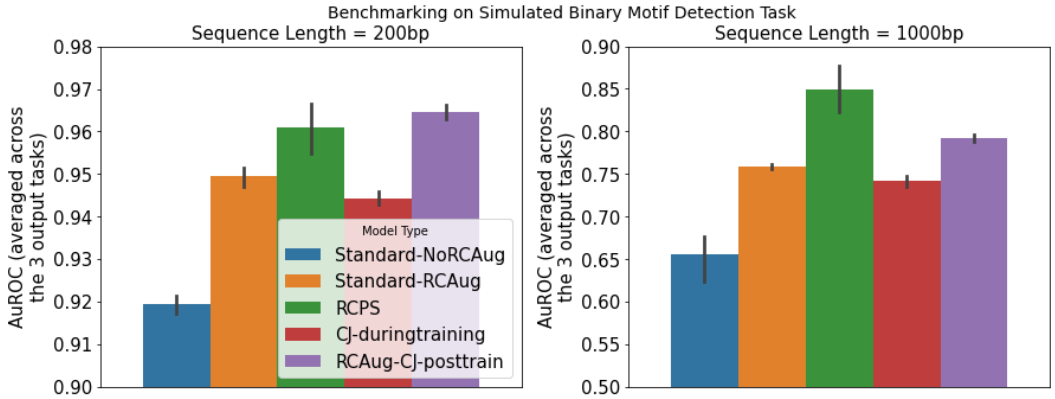

Figure S6: **Benchmarking on simulated motif detection task.** Bar heights represent the average performance over 10 random seeds, and error bars represent the 95% confidence intervals for the mean generated by seaborn [10] using 1000 bootstrapped samples. "Standard-RCAug" and "Standard-noRCAug" are standard models trained with and without RC data augmentation. RCPS denotes RC parameter sharing. CJ-duringtraining are models that were conjoined during training. RCAug-CJ-posttrain are standard models trained with RC data augmentation that were converted to conjoined models after training (i.e. post-hoc conjoined). The best validation-set epoch was selected using AuROC (rather than loss) as the performance metric.

Our processed TF ChIP-seq datasets were identical to those created by Shrikumar et al. [4]. The raw data was produced by the ENCODE consortium [13]; for Ctcf, the specific file used was "wgEncodeBroadHistoneGm12878CtcfStdAlnRep0", for Spi1, the file used was "wgEncodeHaibTfbsGm12878Pu1Pcr1xAlnRep0", and for Max, the file used was "wgEncodeSydhTfbsGm12878MaxIggmusAlnRep0". The positive and negative sets were prepared as follows: for the positive set, we used 1000bp windows centered around the summits of rank-reproducible peaks [14]. For the negative set, 1000bp around the summits of DNase peaks in Gm12878 that did not overlap the top 150K relaxed ChIP-seq peaks of the TF were used (where the "relaxed" peaks were called

by SPP at a 90% FDR). The file used for DNase peaks was "E116-DNase.macs2.narrowPeak.gz", produced by the Roadmap consortium [15]. The training set consisted of all regions except regions on chr1, and chr2, the validation set consisted of regions on chr1, and the testing set consisted of regions on chr2.

S3.1.2 Binary prediction models

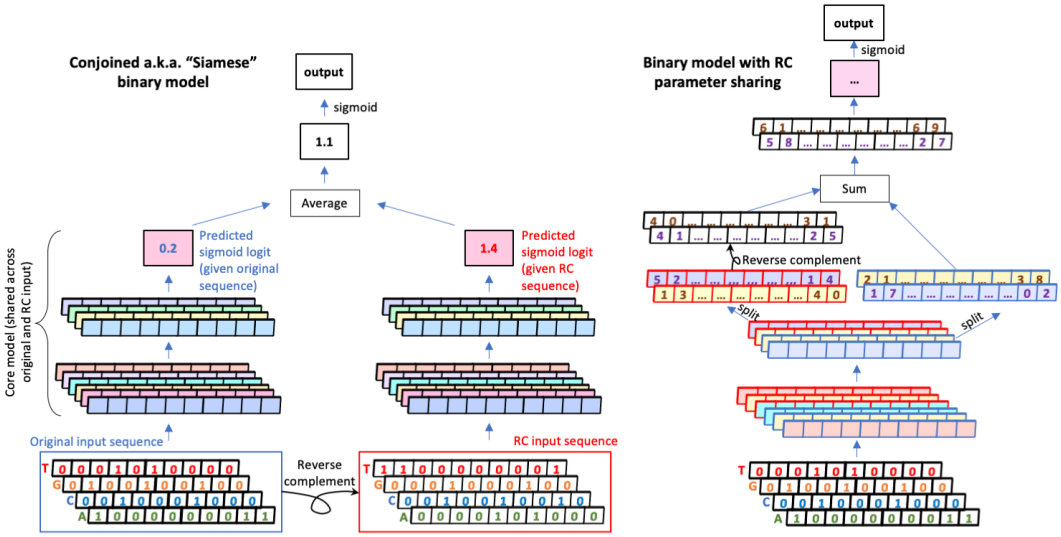

Figure S7: **Architecture of conjoined models and RCPS models for binary prediction**. Cells with similar shading represent convolutional neurons with shared parameters. For the conjoined model, the predictions of the two branches are averaged prior to applying the final sigmoid nonlinearity. In the case of the RCPS models, the representations of the forward and reverse strands are merged by applying the reverse complement operation (i.e. flipping along both the length axis and channels axis) to the output of the last convolutional layer and summing. Blue outlines denote the "forward" versions of a filter, while red outlines denote the corresponding RC versions.

Our model architectures for the binary datasets were adapted from Shrikumar et al. [4]. All standard models trained on the simulated data employed one convolutional layer with 20 filters of kernel width 21 and stride 1, followed by batch normalization and ReLU nonlinearity, followed by maxpooling with width and stride 20, followed by the sigmoid output layer with 3 neurons. All standard models trained on the TF ChIP-seq data had three 16-filter stride-1 convolutional layers of kernel widths 15, 14 & 14 respectively, each of which was accompanied by batch normalization and a ReLU nonlinearity, followed by maxpooling of width 40 & stride 20, followed by the single sigmoid output.

For the conjoined models trained on binary prediction tasks, we averaged the predicted sigmoid logits across both forward and RC inputs prior to passing the result through a sigmoid function to obtain the final prediction. Note that this differs slightly from prior work [5, 3]; in Alipanahi et al. [5], the maximum prediction is taken across strands, while in Quang and Xie [3], predictions are averaged after the sigmoid nonlinearity is applied. We justify taking the average rather than the maximum as this was found to produce superior results as discussed in Bartoszewicz et al. [6], and also because it results in more informative gradient updates for conjoined models during training (if the "max" is applied across both strands, then during training, the gradient of the model with respect to one of the strands would always be zero). We justified taking the average of the sigmoid logits rather than taking the average of the post-sigmoid probabilities because of the equivalence proven in **Sec. S1**, and also because of the following probabilistic interpretation: the logit of a sigmoid represents the log of the predicted odds ratio that the input belongs to the positive class; averaging sigmoid logits can thus be interpreted as taking the geometric mean of two predicted odds ratios. For thoroughness, we compared the performance of conjoined models where the averaging was performance after the sigmoid (as was done in prior work) to when the averaging was performed on the sigmoid logit (as we do here); we found that the mean performance of averaging the logits was slightly higher, but also that there was no significant difference in the means (**Fig. S8**).

For the RCPS architectures trained on binary prediction tasks, we followed Bartoszewicz et al. [6] and summed representations across strands prior to the final sigmoid output layer (**Fig. S7**). This is functionally equivalent to the "weighted sum" layer used in Shrikumar et al. [4] to merge representations, but with a more simplified implementation. Also see the note in **Sec. S3.4** about the effective number of filters present at runtime in RCPS architectures.

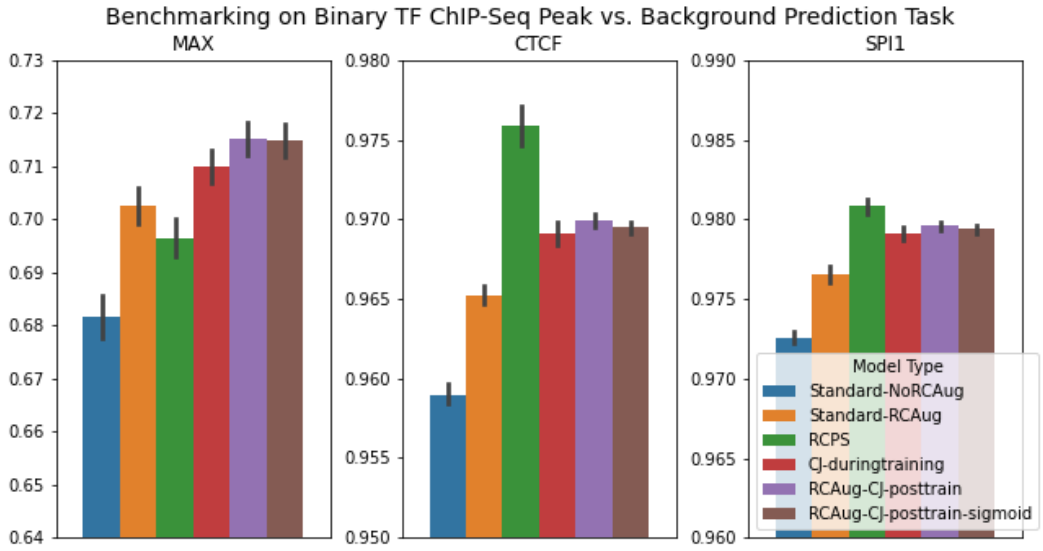

Figure S8: **Comparison of different averaging strategies for the conjoined models trained on binary data.** Bar heights represent the average performance over 10 random seeds, and error bars represent the 95% confidence intervals for the mean generated by seaborn [10] using 1000 bootstrapped samples. "Standard-RCAug" and "Standard-noRCAug" are standard models trained with and without RC data augmentation. RCPS denotes RC parameter sharing. CJ-duringtraining are models that were conjoined during training. RCAug-CJ-posttrain are standard models trained with RC data augmentation that were converted to conjoined models after training (i.e. post-hoc conjoined). RCAug-CJ-posttrain-sigmoid are models with the same architecture as regular RCAug-CJ-posttrain models, but with averaging performed after the sigmoid nonlinearity rather than before it. Similar to **Fig. 2**, training hyperparameters were set to the tested combination that tended to produced the highest overall AuPRCs.

S3.1.3 Binary prediction training hyperparameters

Following Shrikumar et al. [4], all binary prediction models were trained with a binary cross-entropy loss and the Adam optimizer with the default Keras learning rate of 0.001.

In the case of the simulated binary datasets, we replicated the setup from Shrikumar et al. [4] and defined our "epochs" to be 5000 training examples (note: canonically, an "epoch" is defined to be a single pass through the entire training set - however, when the data is loaded on-the-fly using a generator in order to avoid loading all the data at once, epochs can instead be defined by the number of training examples seen). At a batch size of 500 for the simulated data, each "epoch" therefore corresponded to 10 training iterations. The training was terminated when the validation set loss failed to show improvement over 10 consecutive "epochs", and the model at the "epoch" with the best validation set loss was used for performance comparisons.

In the case of the TF ChIP-seq datasets, Shrikumar et al. [4] defined an "epoch" as 5000 training examples, and used batch sizes of 100 - thus, each "epoch" corresponded to 50 training iterations. Shrikumar et al. [4] trained each of the TF ChIP-seq models for 4000 training iterations and selected the model at the epoch that achieved the best validation set AuROC. Shrikumar et al. [4] also upweighted their positive examples according to the class imbalance (16.47:1 for Max, 4.75:1 for Spi1 and 5.01:1 for Ctcf). We replicated this exact setup, but also explored how changes in these training hyperparameters impacted performance (**Fig. S3, S4, S5** and **Sec. S2**) - in particular, we set

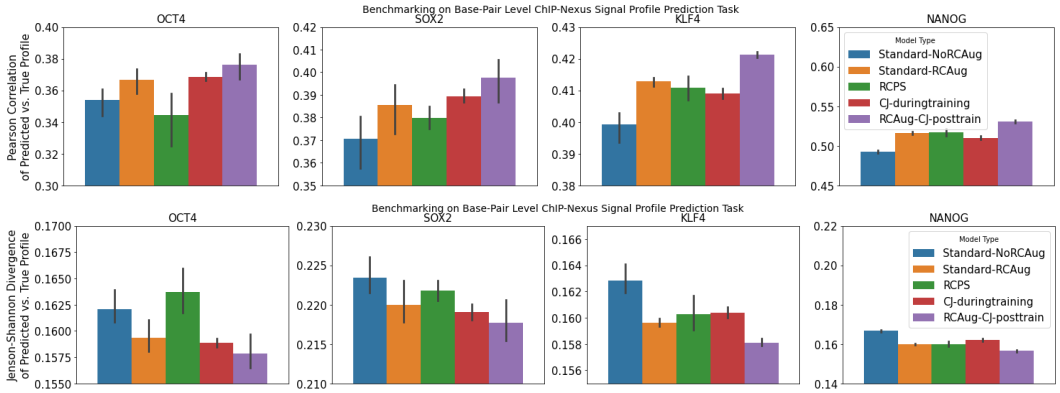

Figure S9: **Benchmarking of profile prediction models - Pearson correlation and Jensen-Shannon divergence.** Spearman correlation metric is in **Fig. 2**. The metrics are explained in **Sec. S3.2.3** Bar heights represent the average performance over 10 random seeds, and error bars represent the 95% confidence intervals for the mean generated by seaborn [10] using 1000 bootstrapped samples. "Standard-RCAug" and "Standard-noRCAug" are standard models trained with and without RC data augmentation. RCPS denotes RC parameter sharing. CJ-duringtraining are models that were conjoined during training. RCAug-CJ-posttrain are standard models trained with RC data augmentation that were converted to conjoined models after training (i.e. post-hoc conjoined).

the maximum number of training iterations to be 8000 (in addition to 4000), we selected the best validation set epoch according to prediction loss (in addition to AuROC), and we handled the class imbalance by up**sampling** positive examples to achieve a 1:4 class ratio (rather than up**weighting** positive examples in the loss).

S3.2 Base-pair-level signal profile prediction

S3.2.1 Profile prediction datasets

We trained our BPNet models using the processed datasets from Avsec et al. [9]. These datasets were ChIP-nexus profiles of the pluripotent TFs Sox2, Oct4, Klf4, and Nanog in mouse embryonic stem cells. At each position and strand in each of the four TFs, the profile included the 5' end read counts. 1000bp windows were selected around the peak summits from the Irreproducible Discovery Rate (IDR) optimal peaks sets [14]. These sequences were inputted into the BPNet Architecture. Because ChIP-nexus experiments can have certain biases, experimental control data was used from PAtCh-CAP59 [16]. The validation set consisted of regions on chr22, the testing set of regions on chr1, chr8, and chr21, and the testing set consisted of all other regions.

S3.2.2 Profile prediction models

BPNet models predict the shape of the observed TF binding signal at base-pair-level resolution in a 1kb interval, using both DNA sequence and the "control" signal track as inputs. The predicted profiles take the form of a probability distribution over each position, for each strand. This probability distribution is compared to the observed distribution of raw read counts in the interval and is scored using a multinomial loss function.

The original BPNet architecture contained two output heads, where one head predicted the shape of the signal profile at base-pair-level resolution, and the other head predicted the total read counts in a given region. For the purpose of controlled benchmarking, we trained models on only the profile prediction head (thereby avoiding differences in performance caused by competition between the two heads). Our BPNet architecture consists of a one-hot encoded input sequence that is supplied to a convolutional layer with 64 filters and a kernel width of 21, followed by a stack of 6 dilated convolutional layers with 32 filters each and a kernel width of 3. The dilation rate of the convolutional layers increased by a factor of 2 with each layer - i.e. the dilations were 2,4,8,16,32 and 64. The output of each layer was also added to the output of the previous layer in order to create the input to subsequent layers. At the end of the stack of dilated convolutional layers (all of which had ReLU

activations), a linear convolutional layer with 2 filters and a filter width of 75 was applied (note that this is equivalent to the "deconvolutional" layer used in the BPNet paper; when the stride is 1, as is the case here, a "deconvolutional" layer is equivalent to a convolutional layer). The output of this layer was then concatenated to the control track's signal profile, and then a second linear convolutional layer with 2 filters and a kernel with of 1 was used to obtain the final profile prediction (one filter each for the positive and negative strands). Note that, from the perspective of the multinomial loss function, these predictions represent the logits of a distribution to which a softmax is implicitly applied (the softmax is applied separately for each strand). All convolutional layers except for the final layer were followed by ReLU nonlinearities. All models were trained using the Adam optimizer and the default Keras learning rate of 0.001. Our handling of reverse-complement symmetry for BPNet-style models is described in **Fig. 1**.

One modification that we made to the BPNet architecture was to avoid zero-padding; in the original BPNet architecture, the inputs to convolutional layers were zero-padded such that the output of the convolutional layer had the same dimensions as its input (this is called "same" padding in keras). The "same" padding was necessary for the residual connections, which perform elementwise additions of different convolutional layers. To avoid zero-padding, we instead supplied a longer initial input sequence (1346bp); in order to make different convolutional layers have compatible dimensions for the residual connections, we trimmed away the ends of the longer layer prior to performing the elementwise addition.

S3.2.3 Metrics to evaluate profile prediction

Profile model predictions were evaluated according to three metrics: Spearman correlation, Pearson correlation, and Jensen-Shannon divergence. Recall that the BPNet's predicted base-resolution signal profiles take the form of a probability distribution. This predicted distribution is compared against the observed distribution of reads. In the case of Spearman and Pearson correlation, we bin both the true distribution and the prediction distribution into bins of size 1bp, 5bp and 10bp, and compute the correlation for each example, strand and binning resolution (here, "binning" means taking the maximum value of the constitutive elements in the bin). In the case of Jensen-Shannon divergence (JSD), the predicted and true probability distributions are smoothed using a Gaussian kernel with $\sigma = 3$, and the JSD is computed separately for each example and strand. The reported values for the Spearman correlation, Pearson correlation and JSD are the average of the correlations across all examples, strands and binning resolutions (if applicable).

S3.3 RC data augmentation

The standard models could be trained with or without RC data augmentation. If RC data augmentation was enabled, it was performed by extending each training batch with the reverse complements of the original inputs, which effectively doubled the batch size.

S3.4 A note on the number of filters in RCPS architectures

In the case of the RCPS architectures, each filter has a reverse-complement counterpart that is created at runtime (via weight sharing), which increases the representational capacity of the model. Thus, the "effective" number of filters of the RCPS model could be considered to be twice the number of filters of the non-RCPS models. To account for this, we also ran benchmarks where the RCPS models were specified to have half the number of filters (such that the "effective" number of filters would be comparable to the number in the non-RCPS architectures).

For the TF ChIP-seq binary datasets, the RCPS models with half the number of filters had lower AuPRCs than their counterparts with the full number of filters (**Fig. S10**). This difference was most apparent in the Max TF ChIP-seq binary task, where the reduction in filters caused the RCPS models with half the number of filters to perform considerably worse than all other models, including standard models trained without data augmentation. These trends are consistent with other anomalies observed on the Max TF ChIP-seq dataset (**Sec. S2**). For Ctcf, the performance of RCPS models with half the number of filters was still superior to the other non-RCPS models benchmarked, and for Spi1 TF-ChIP-seq the RCPS models with half filters retained comparable performance to the best non-RCPS models.

For the base-pair-level signal profile prediction datasets, the mean performance of the half-filter RCPS models was again always lower than that of the full-filter RCPS models (**Fig. S11**). This trend was observed across all three different profile prediction metrics (Spearman correlation, Pearson correlation, and Jensen-Shannon divergence). For Sox2 and Klf4, we found that the 95% confidence interval for the mean performance over 10 random seeds overlaps with the confidence interval of standard model trained without data augmentation (confidence intervals were generated by seaborn [10] via bootstrapping). This poor performance of the half-filter RCPS models is consistent with the generally worse performance of RCPS architectures on profile prediction tasks.

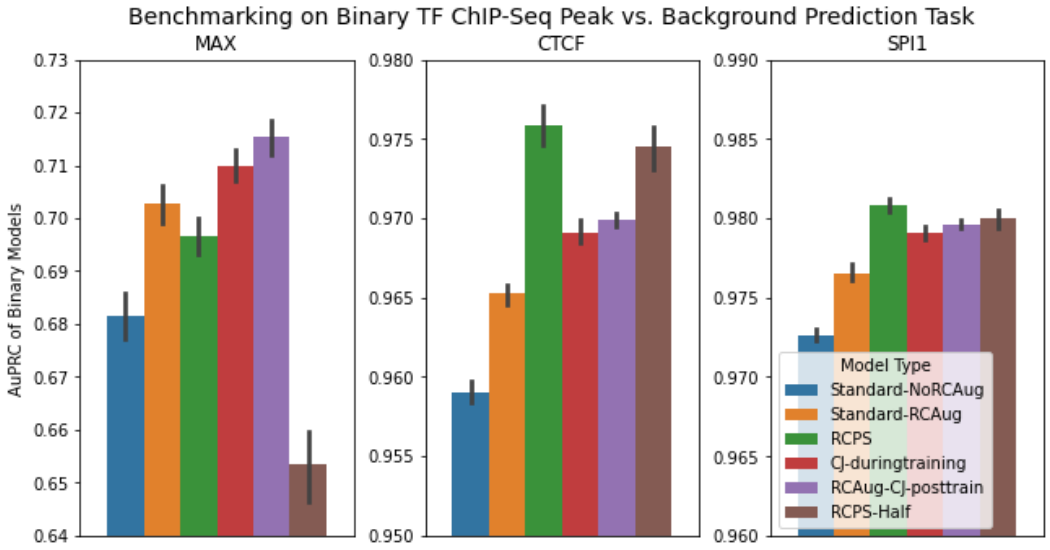

Figure S10: **Performance of RCPS models with half the number of filters on binary TF ChIP-Seq datasets.** Bar heights represent the average performance over 10 random seeds, and error bars represent the 95% confidence intervals for the mean generated by seaborn [10] using 1000 bootstrapped samples. "Standard-RCAug" and "Standard-noRCAug" are standard models trained with and without RC data augmentation. RCPS denotes RC parameter sharing. CJ-duringtraining are models that were conjoined during training. RCAug-CJ-posttrain are standard models trained with RC data augmentation that were converted to conjoined models after training (i.e. post-hoc conjoined). RCPS-half are models with the same architecture as regular RCPS models but with half the number of filters. Similar to **Fig. 2**, training hyperparameters were set to the tested combination that tended to produced the highest overall AuPRCs.

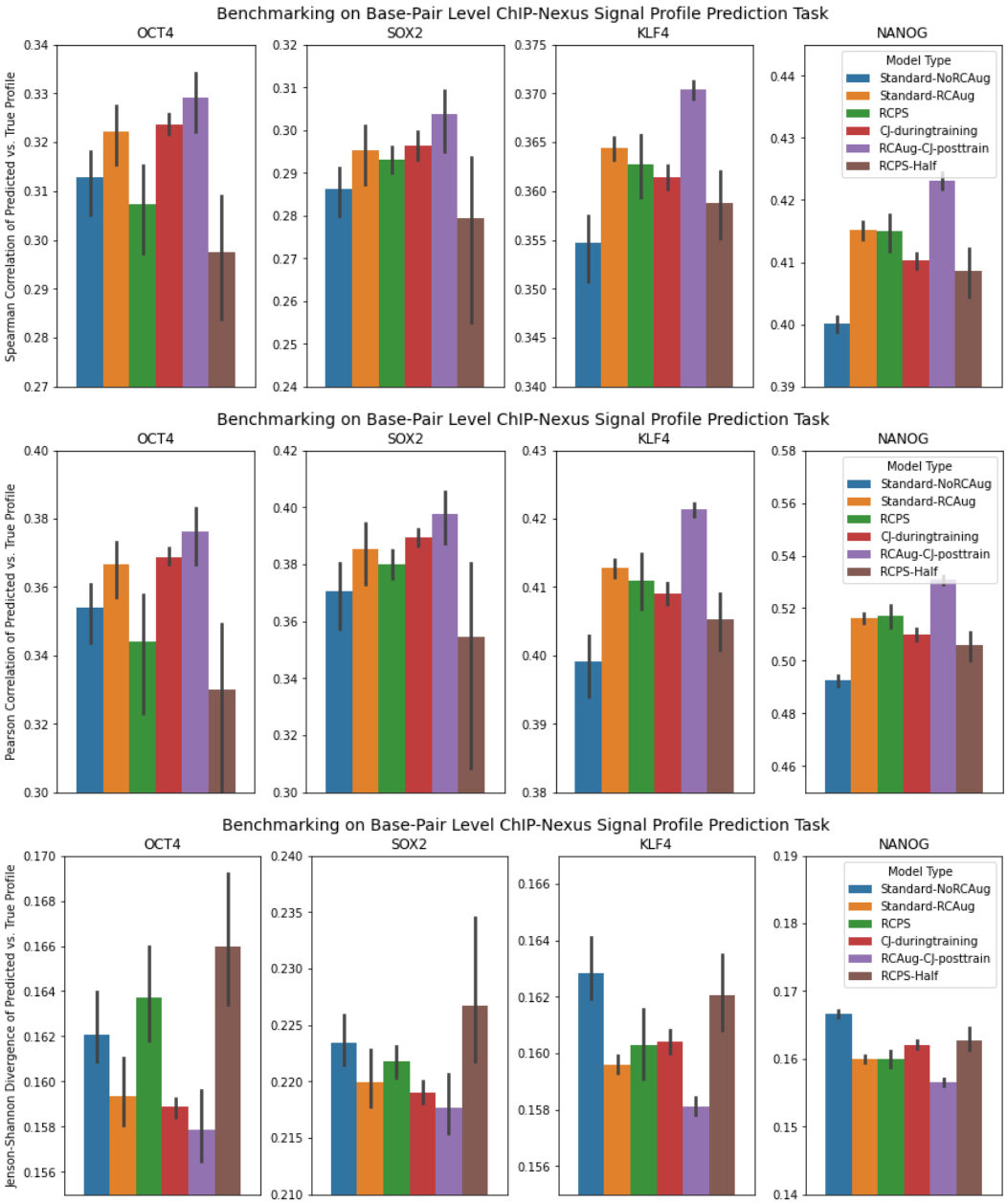

Figure S11: **Performance of RCPS models with half the number of filters on profile prediction datasets.** Bar heights represent the average performance over 10 random seeds, and error bars represent the 95% confidence intervals for the mean generated by seaborn [10] using 1000 bootstrapped samples. "Standard-RCAug" and "Standard-noRCAug" are standard models trained with and without RC data augmentation. RCPS denotes RC parameter sharing. CJ-duringtraining are models that were conjoined during training. RCAug-CJ-posttrain are standard models trained with RC data augmentation that were converted to conjoined models after training (i.e. post-hoc conjoined). RCPS-half are models with the same architecture as regular RCPS models but with half the number of filters.