Methods to improve prediction of gene expression from noncoding regulatory DNA sequence

by

Kanav Mittal

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Bachelor of Science

in

Electrical Engineering and Computer Science

in the

Undergraduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Nilah Ioannidis, Chair

Spring 2025

Methods to improve prediction of gene expression from noncoding regulatory DNA sequence

Abstract

Methods to improve prediction of gene expression from noncoding regulatory DNA sequence

by

Kanav Mittal

Bachelor of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Nilah Ioannidis, Chair

Numerous deep learning models have been recently developed to predict gene expression from the surrounding DNA sequence which includes regulatory elements. While most models jointly predict both chromatin regulation and gene expression data from sequence, the ExPecto framework uses a hierarchical approach, first using a convolutional neural network (CNN) to predict chromatin features from sequence and then a linear model to predict gene expression from chromatin features. In this work, we develop and assess CNN hierarchical models to predict gene expression from chromatin features. We find that our CNN models outperform linear models and baselines and perform similarly to finetuning Enformer, and we find that converting the non-hierarchical Enformer into a hierarchical model improves gene expression predictions. These findings suggest that hierarchical approaches warrant further investigation for their potential to improve predictive accuracy. We next analyze the performance of the hierarchical approach and Enformer-finetuning approach on predicting celltype-specific expression values. We find that performance improves by using a loss function that explicitly considers expression values across all cell types. We finally find that finetuning Enformer with this loss function does not significantly change how much attention is paid to distal bins, showing that a cross-celltype loss does not help a model better integrate long-range signal from distal enhancers.

Contents

List of Figures

List of Tables

Acknowledgments

Thank you to Professor Nilah Ioannidis for providing so much guidance on this project. Thank you for taking me into your lab over two years ago. I have learned so much here, not just about regulatory genomics, but also about how to be a good researcher.

Thank you to my mentor Ruchir Rastogi for all your help and support. Thank you for teaching me about the field and always being available to answer any of my questions. I have especially enjoyed being able to brainstorm with you.

Finally, thank you to my family and friends for always supporting me through my academic journey. Special shoutout to my friend Moe Putterman for providing emotional support as I tried to figure out how to finetune a model.

Chapter 1

Introduction

1.1 Regulating gene expression

Proteins consist of a remarkably diverse group of macromolecules responsible for various processes essential for life. Cells synthesize proteins according to the instructions encoded in DNA sequences called genes. A gene is said to be expressed when it is used to produce proteins (and non-coding RNAs), and these proteins eventually influence the phenotype of a cell or organism. Crucially, genetic information containing instructions to synthesize proteins does not flow directly from DNA to protein; rather, the central dogma of molecular biology states that genetic information flows from DNA to RNA to protein [5]. In particular, messenger RNA (mRNA) acts as a carrier of genetic information from DNA to protein [8]. mRNA is synthesized from DNA by the RNA polymerase enzyme, in a process known as transcription.

Complex organisms consist of hundreds of different cell types, each of which has a specialized function that requires a different set of proteins. (Note that in this work, we use tissue and cell type mostly interchangeably.) Because all cells contain virtually the same DNA sequences, regulating gene expression is key for cells to be able to produce only the proteins necessary for that cell's survival and role [3].

Around 98% of the human genome is made up of noncoding DNA, and much of this noncoding DNA plays a role in regulating gene expression. Important cis-regulatory elements (CREs) in the noncoding genome include promoters, which are short sequences of DNA near a gene's transcription start site (TSS), and enhancers, which can be found further away. A distal enhancer, in particular, refers to an enhancer far from the TSS. Transcription factors (TFs) are proteins that modulate transcription initiation via the binding of RNA polymerase to DNA. CREs modulate transcription by interacting with various TFs [6].

Because transcription involves the binding of RNA polymerase to DNA, DNA accessibility, also known as chromatin accessibility, is an important factor in regulating gene expression. DNA is packaged with histones and other proteins in a complex known as chromatin. Chemical modifications to histone proteins affect the structure of chromatin, and

thereby the accessibility of DNA [13].

Sequencing methods

A number of methods have been developed to study CREs, chromatin accessibility, and gene expression. These methods include

Method	**Used to measure**
DNase-seq	Chromatin accessibility
ATAC-seq	Chromatin accessibility
ChIP-seq	TF binding and histone modifications
CAGE-seq	Gene expression
RNA-seq	Gene expression

Table 1.1: Summary of sequencing methods and what they measure.

These methods have been used by large-scale experiments to study different regulatory elements, histone modifications, and gene expression in hundreds of different cell types [6].

1.2 Deep learning models for regulatory genomics

The regulatory grammar that explains how CREs interact with each other to influence gene expression is remarkably complex. Fortunately, deep learning models for regulatory genomics have been shown to be successful in capturing this complexity. These models are crucial to interpreting the noncoding genome and predicting how noncoding DNA sequence affects gene expression.

The success of these models is in part due to their expressivity: having multiple layers makes the models well-suited to process complex information and understand the interactions between regulatory elements in different parts of a sequence. Furthermore, numerous innovations in deep learning have become popular choices for the architectures of these regulatory genomics models, including convolutional neural network (CNN) layers—used in models such as DeepSEA [17], Basset [11], and Basenji [12, 10]—and Transformer or attention layers—used in models such as Enformer [2] and Borzoi [15].

The success of these architectural choices can be partially explained by their ability to effectively model underlying biological mechanisms. As evidenced by the weights in their first layer, CNNs have been shown to learn to recognize sequence motifs for transcription-factor binding sites [11]. Additionally, Transformers are used to integrate information across a longer context size, particularly from enhancers that are considered distal i.e., far from the transcription start site (TSS) of the gene [2].

Many models for regulatory genomics are considered sequence-to-expression models, meaning that they take in DNA sequence and output predictions for (tissue-specific) gene

expression. The DNA sequence is often centered at the TSS and contains some of the surrounding regulatory DNA sequence. Models vary significantly in the length of input DNA sequence they can take in, also known as the model's context size.

Chapter 2

Hierarchical modeling for sequence-to-expression prediction

Much of this chapter comes from work that was presented at the 2024 Machine Learning in Computational Biology conference: "Exploring hierarchical model frameworks for predicting gene expression from sequence" by Kanav Mittal, Ruchir Rastogi, and Nilah Ioannidis

2.1 Hierarchical modeling

To better capture regulatory grammar, sequence-to-expression models are often trained not only to predict gene expression, but also chromatin regulation data, such as transcription factor binding, DNA accessibility, and histone modifications. Two main architectural approaches have emerged to jointly incorporate gene expression and regulatory data. The first approach, epitomized by Basenji, Enformer, and Borzoi, predicts both gene expression and regulatory data in a multitask fashion. Conversely, ExPecto [18] proposes a hierarchical strategy where DNA sequence is used to predict regulatory data and the predicted regulatory data is used to predict gene expression. In this work, we primarily consider the ExPecto and Enformer sequence-to-expression models.

ExPecto

In ExPecto, a CNN called Beluga (a successor to DeepSEA) takes in a one-hot encoded DNA sequence of length 2 kb and outputs predictions for 2002 regulatory chromatin features for the center 200 bp of this 2 kb sequence. The 2002 features include DNase-seq, TFs, and histone modification tracks sourced from ENCODE [4] and Roadmap Epigenomics [14]. ExPecto repeatedly applies Beluga to the 40 kb sequence centered at the TSS of a gene to come up with predictions consisting of 200 bins, each representing 200 bp. The predicted chromatin features are then integrated using predefined spatial basis functions and input into tissue-specific regularized linear models to predict gene expression.

Enformer

Enformer takes in a one-hot encoded DNA sequence of length 196,608 bp centered at the TSS and outputs predictions for 4675 human regulatory chromatin features and 638 human CAGE expression measurements across the center 114,688 bp of the input sequence, binned in 128 bp bins. The 4675 features include DNase-seq, TF via ChIP-Seq, histone modification via ChIP-Seq, and ATAC-seq tracks sourced from ENCODE and Roadmap Epigenomics. The 638 CAGE expression measurements are sourced from FANTOM5 [1]. Enformer uses CNN and Transformer layers to make predictions for both human and mouse tracks, and the model was trained with both human and mouse data. In this work, we only consider the human predictions made by Enformer.

Introduction to the hierarchical approach

The hierarchical approach exemplified by ExPecto, which decomposes the sequence-to-expression prediction problem into a sequence-to-chromatin module and a chromatin-to-expression module, has several potential advantages. First, it may better approximate causal biological mechanisms. Second, we hypothesize that predicting chromatin features requires only local DNA sequence context, whereas predicting gene expression requires a broader context. Attempting to predict both simultaneously may lead to negative transfer, potentially explaining previous findings that multitask models struggle to capture distal regulatory effects [9]. Third, having two distinct modules allows users to train custom chromatin-to-expression models on new gene expression datasets without retraining the sequence-to-chromatin model.

Despite these potential advantages, there is limited follow-up work exploring architectural improvements to hierarchical modeling of gene expression. In this work, we introduce two new chromatin-to-expression architectures: single and multi-tissue CNNs. We compare their performance to the linear model proposed in ExPecto across many settings, including using experimental versus predicted regulatory chromatin features as inputs, using predicted regulatory chromatin features from Beluga or Enformer, and using regulatory chromatin features from different genomic context sizes. We provide an overview of our hierarchical modeling setup in Figure 2.1.

Methodology

Data

We trained chromatin-to-expression models to predict expression measurements for 24,336 genes across 218 tissue samples that were collected by the Genotype-Tissue Expression Project (GTEx) [16], Roadmap Epigenomics, and ENCODE. Expression measurements were reported as log reads per kilo base per million mapped reads (RPKM) RNA-seq values, as preprocessed by [18]. For our analysis, we excluded genes that coded for ribosomal RNA (rRNA), leaving us with 22,828 genes that coded for either proteins or long intergenic noncoding RNAs.

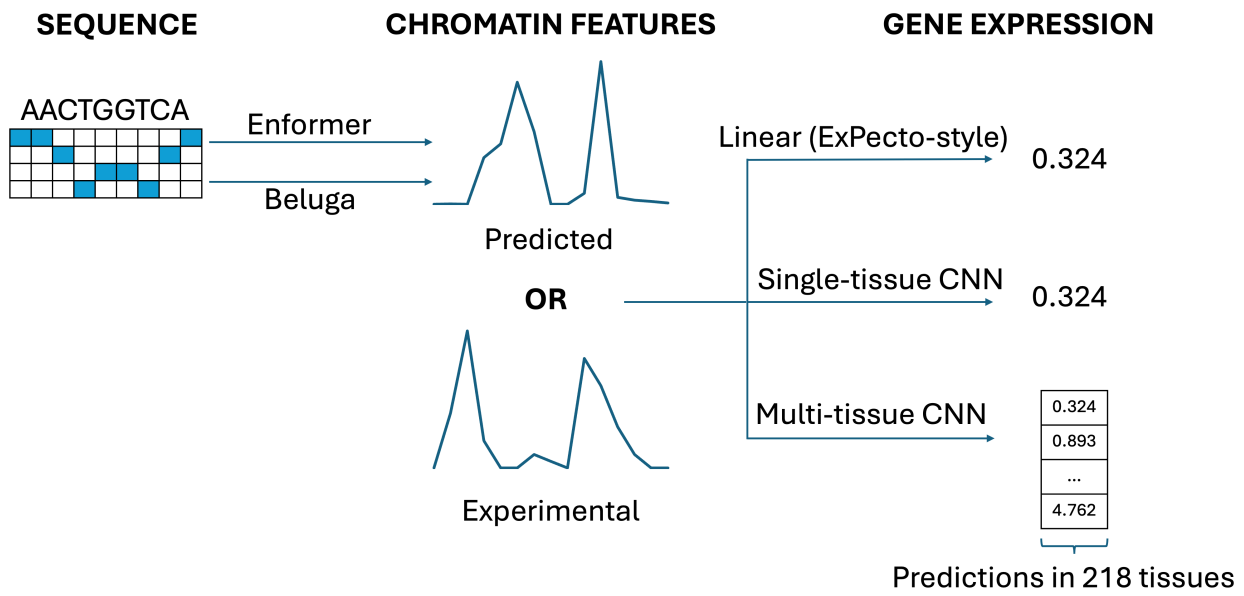

Figure 2.1: **Overview of the hierarchical modeling strategy.** Enformer and Beluga are used to predict chromatin regulation features from DNA sequence. These predictions (or experimental chromatin regulation features) are input to ExPecto-style tissue-specific linear models, single-tissue CNNs predicting tissue-specific gene expression, or multi-tissue CNNs predicting gene expression over all 218 tissues.

For models using Beluga predictions (or Beluga experimental data) as input, we tested on genes on chromosome 8 because Beluga was not trained on that chromosome. Similarly, for models using Enformer predictions (or Enformer experimental data), we tested on genes on chromosome 14.

To construct Beluga predictions for each gene, we took the sequence centered at the TSS of that gene from the GRCh37 human reference genome and repeatedly passed 2 kb sequence windows into the Beluga model for a total of 200 times to construct Beluga predictions with 200 bins. To construct Enformer predictions for each gene, we took the 196,608 bp sequence centered at the TSS of that gene from the GRCh38 human reference genome and passed this into the Enformer model. Because [18] gave TSSs in GRCh37 coordinates, we converted these to GRCh38 coordinates for the Enformer predictions. We used the "CAGE_representative_TSS" as defined by [18] for all the TSSs.

To construct Beluga experimental data, we obtained bigWig files for 1158 of the 2002 Beluga training tracks from ENCODE and Roadmap Consortium. We averaged values in 200 bp bins for the 200 bins surrounding the TSS of each gene. (Note that we used all 2002 tracks when using predicted Beluga features as input, not just the 1158 for which we found experimental data.) For Enformer experimental data, we used preprocessed data from [10], which aggregates base pair data into 128 bp bins. To approximate experimental values for

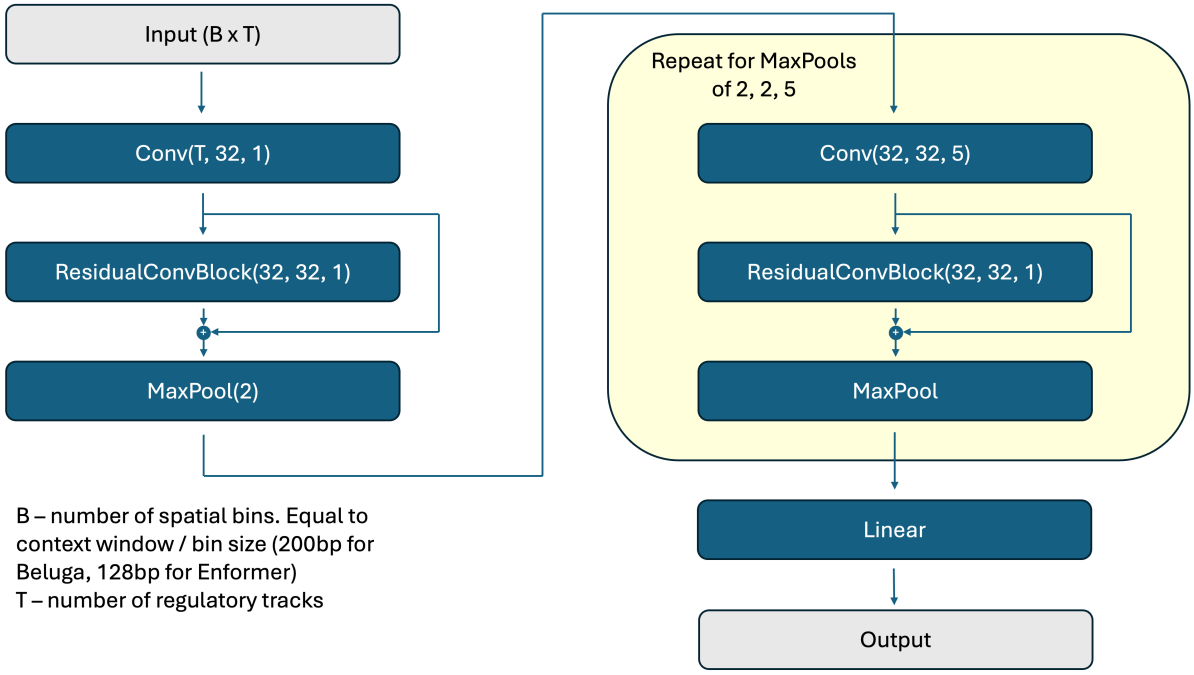

Figure 2.2: **Architecture of the hierarchical CNN**. Conv represents a convolutional layer; ResidualConvBlock represents a batch normalization, Gaussian Error Linear Unit (GELU) activation function, and a convolutional layer with a residual connection around the block; MaxPool represents a max pooling layer; and Linear represents a linear layer.

the 896 bins of 128 bp surrounding the TSS of each gene, we averaged the values of all preprocessed bins that overlapped with these bins. Unless otherwise specified, for models using Enformer predictions or experimental data, we included only the 4675 non-CAGE human tracks as input.

Models

We trained three different chromatin-to-expression models: (1) tissue-specific ridge regression models that use the same spatial basis functions for dimensionality reduction as described in [18], (2) single-tissue CNNs predicting tissue-specific gene expression, and (3) multi-tissue CNNs predicting gene expression in all 218 tissues simultaneously. Both CNN models consist of 8 convolutional layers, interspersed with residual connections and max pooling layers to reduce the spatial dimension. The architecture of the CNN is described in Figure 2.2; the final output linear layer outputs 1 value for single-tissue CNNs and 218 values for multi-tissue CNNs. Grid search on validation set performance (validation cross-gene Spearman correlation) was used to determine the regularization parameter for ridge models and the learning rate and dropout probability for CNNs. We evaluated the predictions of our model

on the cross-gene task, which means that we evaluate how well the model can order the expression values of the genes in our test set. We use Spearman rank correlation to quantify this.

Results

Performance with experimental chromatin features as input.

To determine whether more complicated architectures can improve on the linear chromatin-to-expression model proposed in ExPecto, we first trained single-tissue and multi-tissue CNNs on experimental chromatin regulation data. We sourced experimental data from the Beluga and Enformer training datasets. Both datasets contain transcription factor binding, chromatin accessibility, and histone mark assays from ENCODE and Roadmap Epigenomics, but the Enformer dataset contains many more experiments (4675 for Enformer, compared to the 1158 for Beluga that we were able to obtain).

In both the Beluga and Enformer, we find that CNNs outperform linear models (Fig. 2.3). Single-tissue CNNs using the Beluga dataset improve performance from 0.862 to 0.884 (2.5%), and multi-tissue CNNs using 114.7 kb of context from the Enformer dataset improve performance from 0.830 to 0.838 (1.0%). To explore whether chromatin-to-expression model performance depends on the context size of chromatin features included, we also trained linear and multi-tissue CNN models on the Enformer dataset with a much shorter 40.2 kb context size. We find improved performance of both the linear model and the multi-tissue CNN with the shorter context size, indicating that models of various complexity struggle to capture the effects of distal regulatory elements in concordance with the findings of [9]. Despite our finding that performance improves with a shorter context size, we work with the full context size unless otherwise specified for the rest of this work as we hope that other modeling strategies can better capture the effects of distal regulatory elements. (Note that our results should not be interpreted to indicate that the experimental chromatin regulation features in the Beluga dataset are more predictive of gene expression than the features in the Enformer dataset, as we tested the models on genes in different chromosomes.)

Performance with predicted chromatin features as input.

We then checked if the above performance improvements were replicated when using predicted, as opposed to experimental, chromatin features from Beluga and Enformer as input. We find that a linear model slightly outperforms both single-tissue and multi-tissue CNNs when using predicted Beluga features as input (Fig. 2.4), suggesting that a simpler model does better when there are sufficient errors in the input. However, CNNs outperform linear models when using predicted chromatin features from Enformer: single-tissue CNNs improve performance by 1.3% compared to linear models when using 114.7 kb of context, and multi-tissue CNNs improve performance by 2.1% compared to linear models on the shorter 40.2 kb context length.

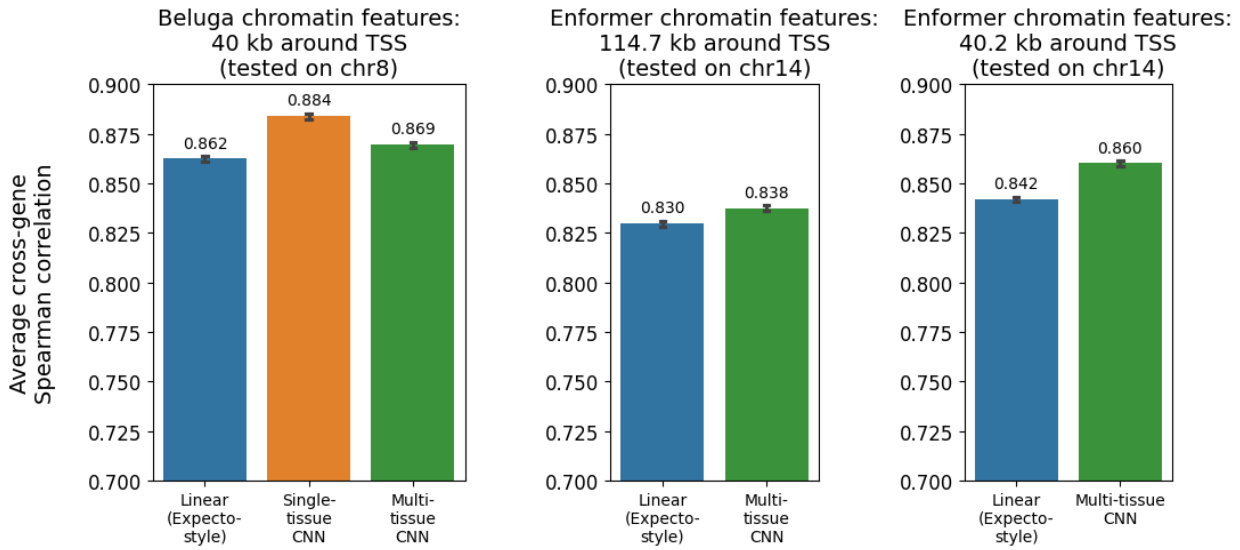

Figure 2.3: **Performance of chromatin-to-expression models on cross-gene task using experimental chromatin features as input.** We train and evaluate on the experimental chromatin features used for Beluga and the experimental chromatin features used for Enformer. We calculate the cross-gene Spearman correlation between expression predictions and experimentally measured values across all genes in each of the 218 cell types. We plot the average correlation across all cell types, with error bars reflecting ±1 standard error of the mean.

We observe that there is little change in performance when using experimental chromatin features from Enformer's dataset as opposed to Enformer model predictions (compare Fig. 2.3 to Fig. 2.4), indicating that Enformer has a strong understanding of chromatin regulation. This is, however, not true for Beluga.

2.2 Benchmarking the hierarchical modeling approach

Models like Enformer innately incorporate signal from distal regions of sequence. Because the use of distal information helps with tissue-specific gene expression prediction [7], Enformer is a good candidate model that we can apply to our downstream gene expression task. We experiment with finetuning Enformer on our dataset and building linear probes on Enformer embeddings or outputs, and we use the performance of these methods to benchmark our hierarchical modeling approach.

We compared the performance of our hierarchical modeling strategy with baseline linear probing models. We build three different linear probes: linear probing on outputs, subset to just CAGE tracks, linear probing on outputs with all tracks, and linear probing on embeddings.

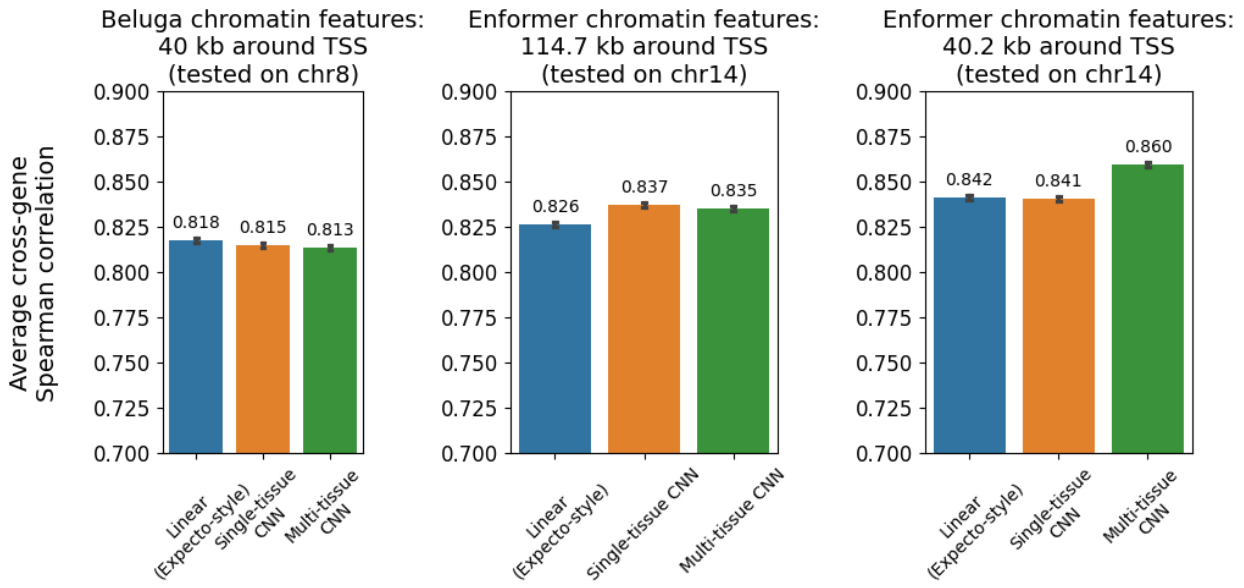

Figure 2.4: **Performance of chromatin-to-expression models on cross-gene task using predicted chromatin features as input.** We calculate the cross-gene Spearman correlation between expression predictions and experimentally measured values across all genes in each of the 218 cell types. We plot the average correlation across all cell types, with error bars reflecting ±1 standard error of the mean.

We also compared the performance of our hierarchical modeling strategy with finetuning Enformer. Finetuning Enformer differs from building a hierarchical model on top of Enformer predictions in two key ways:

- *Output heads*: Hierarchical modeling involves training chromatin-to-expression models on top of the outputs of Enformer i.e., Enformer's output heads are involved in the final expression predictions. However, finetuning Enformer involves building a task-specific head or heads on top of the embeddings of Enformer i.e., Enformer's output heads are not involved in the final expression predictions.
- *Parameter updates*: With hierarchical modeling, we only update the parameters in our chromatin-to-expression models. With finetuning, we have the option to update all the parameters: the parameters of both Enformer and of the task-specific head(s) we build. As a result, finetuning is potentially more computationally expensive.

Methodology

Linear probing baselines

For each gene in our dataset, we pass the complete sequence of 196,608 bases centered at the TSS into Enformer. Depending on the type of linear probe, we take just the output CAGE tracks, all output tracks, or the embeddings of the model. We build three different linear probes using different sources of input data.

- Linear probing on output CAGE tracks. The output tracks of Enformer include 638 human CAGE tracks that capture cell-type-specific expression across a different set of tissues and cell types than our dataset of 218 tissues and cell types. From each of the outputs, we extract the 638 human CAGE tracks.
- Linear probing on outputs. From each of the outputs, we use all 5313 human tracks.
- Linear probing on embeddings. From the Enformer model, we extract the embedding (dimension 3072) right before the output heads.

The outputs and embeddings of the model are across 896 spatial bins. To reduce the size of our linear regression probing models, we average the center 10 bins (i.e., the 10 bins around the TSS). We then fit 218 cell-type-specific linear regression models that take in these bin-averaged values and output the corresponding cell-type-specific gene expression. We train on our 21988 train sequences and evaluate on our 420 test sequences.

Finetuning Enformer

We finetuned Enformer on the cross-gene task, using our dataset of celltype-specific expression across 218 cell types. We did not finetune Beluga on the cross-gene task with our dataset. Beluga makes a prediction in a 200 bp bin using only the surrounding 2 kb sequence, meaning that Beluga does not integrate signal from distal regions of the sequence. Because distal information is important for tissue-specific gene expression prediction [7], we doubt that Beluga would be a good candidate for finetuning on the task of predicting gene expression.

We pass in the human reference genome sequences corresponding to the genes in our dataset: 21988 train sequences, 420 validation sequences, and 420 test sequences. Due to compute restrictions, we only pass into Enformer the center 50kb of each of these sequences, rather than the 196,608-length sequence that Enformer can take in. For each gene sequence, we extract the embedding right before the output heads. We then extract the center 10 out of the 391 bins in the embedding, we apply an attention pool layer to aggregate information across the center 10 bins using learnable weights for each bin, and we finally send this aggregated embedding through a linear layer to predict gene expression across 218 cell types.

For each gene sequence, we now have predicted cell-type-specific gene expression values. We compute Mean Squared Error loss with our experimentally-measured cell-type-specific

gene expression values. We do not freeze any parameters i.e., we update all parameters according to the gradient of the loss. Due to limited compute, we did not perform hyperparameter optimization. The hyperparameters chosen based on past experiments included a learning rate of 0.0001, weight decay of 0.001, and a batch size of 2. The batch size is quite low due to compute restrictions. We trained for 50 epochs with a patience of 5 epochs on validation loss. Finally, we used the Linear Warmup Cosine Annealing learning rate scheduler with 1000 warmup steps.

Results

Linear probing baselines

According to Figure 2.5, we find that linear probing on embeddings outperforms linear probing on the output tracks, both when we include all output tracks and when we subset to just CAGE tracks. This implies that (for the center 10 bins) the embedding space of the Enformer model is more informative and predictive of gene expression than the outputs of Enformer. Since the embeddings of Enformer are passed through a linear transformation to produce the outputs, the outputs can contain no more information than the embeddings themselves. We also do find that linear probing on all outputs tracks outperforms linear probing on just the output CAGE tracks. This indicates that the other, non-CAGE tracks in the Enformer outputs contain useful signal for predicting gene expression.

We do find that all of the CNN hierarchical models (single and multi-tissue CNN) outperform all of the baselines, as seen in Figure 2.5. The linear, ExPecto-style hierarchical model outperforms both the baselines that involve linear probing on Enformer outputs (both all output tracks and just CAGE tracks). As shown in [9], Enformer struggles to properly account for distal information. This phenomenon affects the performance of the linear probing baselines, which only consider the central ten bins around the TSS. However, hierarchical models give us the opportunity to recapture some of the distal information that was previously ignored by the Enformer model. The CNN hierarchical models use convolutions and pooling to integrate signal from across bins, and the linear, ExPecto-style models use the predefined spatial basis functions to achieve this. Predictions for gene expression are based more on distal information than predictions for regulatory chromatin features, and, as a result, hierarchical models help us to make better predictions for gene expression. This suggests that prediction performance might generally be improved by transforming a non-hierarchical model into a hierarchical model.

Finetuning Enformer

We find comparable performance on the cross-gene task between the hierarchical modeling strategy, for both single and multi-tissue hierarchical CNNs, and finetuning Enformer. We additionally find that finetuning Enformer outperforms all baselines. However, our strategy to finetune Enformer was quite computationally expensive. We explore freezing parameters

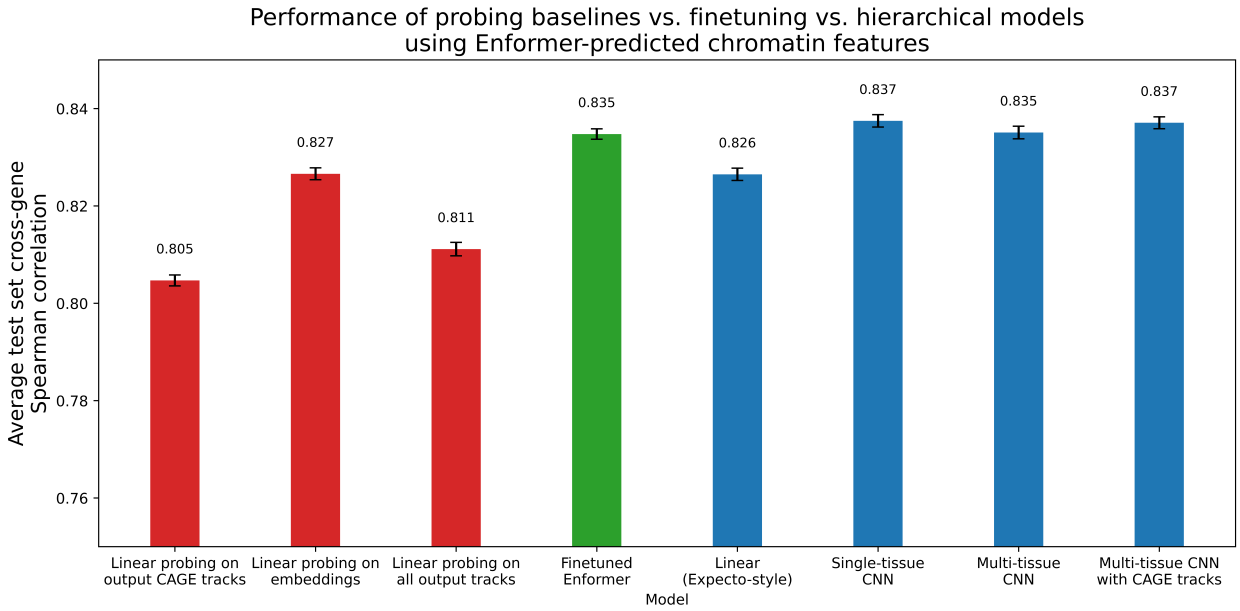

Figure 2.5: **Performance of linear probing baselines versus finetuning versus hierarchical models on cross-gene task using Enformer-predicted chromatin features as input.** Red indicates linear probing baselines, green indicates finetuned Enformer, and blue indicates hierarchical models. We calculate the cross-gene Spearman correlation between expression predictions and experimentally measured values across all genes in each of the 218 cell types. We plot the average correlation across all cell types, with error bars reflecting ± 1 standard error of the mean.

when finetuning to reduce cost in Chapter 3, and other strategies should be explored in future work.

2.3 Hierarchical modeling with CAGE tracks in the input

Methodology

We repeat a similar process as training the hierarchical chromatin-to-expression models described earlier. The previous hierarchical models did not include the 638 Enformer CAGE tracks in their input. However, now we train hierarchical models on top of Enformer predictions and experimental data that include the 638 Enformer CAGE tracks as part of the input data. These tracks do not represent chromatin features but rather represent expression values measured with CAGE-seq. Meanwhile, the 218 celltype-specific expression values we are trying to predict come from RNA-seq. The other slight difference in methodology is

that while we earlier trained the hierarchical models without CAGE tracks in input for a maximum of 5 epochs, these hierarchical models with CAGE tracks in input are trained for a maximum of 10 epochs.

Results

As shown in Figure 2.5, we get similar performance when we include or when we don't include CAGE tracks in the input to the hierarchical multi-tissue CNN. It appears that there is enough information within the non-CAGE predictions of Enformer to make gene expression predictions (RNA-seq) that adding CAGE tracks does not help us more. We hypothesize that if we added ground-truth experimental CAGE tracks into the input to the model, we would see improved performance. This is because both CAGE and RNA-seq are measurements of gene expression, so having experimental CAGE measurements should help the model better predict RNA-seq measurements. However, Enformer's predictions for CAGE tracks contain enough noise that they do not help make better RNA-seq predictions. On the other hand, we believe that the non-CAGE predictions of Enformer (i.e., the TF binding, histone modification, and chromatin accessibility) tracks are less noisy. This difference between non-CAGE and CAGE output tracks may be attributed to the fact that the non-CAGE predictions do not require as much distal context as CAGE predictions [7], and we know that Enformer struggles with integrating information from distal context [9].

2.4 Conclusion

In this chapter, we explore the hierarchical modeling strategy for sequence-to-expression predictions, inspired by [18]. We evaluate the performance of linear, ExPecto-style models, single-tissue CNN models, and multi-tissue CNN models on the cross-gene task, finding that CNNs outperform linear models on Enformer predictions. Additionally, we compare our hierarchical modeling approach to two benchmarks: a finetuned Enformer and linear probing baselines built on Enformer predictions. We find that our hierarchical strategy performs comparably to finetuned Enformer and performs at least as good as the baselines, showing that the hierarchical approach is promising. We also find that including predicted CAGE tracks in the input to hierarchical models does not help performance. The hierarchical approach presents numerous benefits and deserves further exploration. We believe that the optimal architecture for a hierarchical model depends on the complexity of the task, and we leave it to future work to determine the best hierarchical architecture for a number of downstream tasks.

Chapter 3

Improving the cell-type specificity of sequence-to-expression models

3.1 Hierarchical modeling and cross-celltype loss

Performance on cross-celltype task

The cross-celltype task tests whether a model captures the differences in expression for one gene across multiple tissues. We chose to measure performance on the cross-gene task by looking at whether the model can, for one gene, correctly order the expression values across the 218 tissues in our dataset, as calculated by the Spearman rank correlation coefficient. When measuring cross-celltype performance, we average cross-celltype Spearman correlation across all genes in our test set.

Methodology

We train hierarchical multitask CNNs (i.e., same as multi-tissue) on top of four different types of input data. For Enformer input data, we only consider non-CAGE tracks.

- Enformer Predictions: the predicted (non-CAGE) output tracks of Enformer
- Enformer Experimental: the experimental (non-CAGE) tracks used to train and evaluate Enformer
- Beluga Predictions: the predicted output tracks of Beluga
- Beluga Experimental: the experimental tracks used to train and evaluate Beluga

We train each model with MSE loss. We perform grid search over learning rate and dropout probability, and we choose the best performing hyperparameters based on validation loss. We evaluate the cross-gene and cross-celltype performance on the test set.

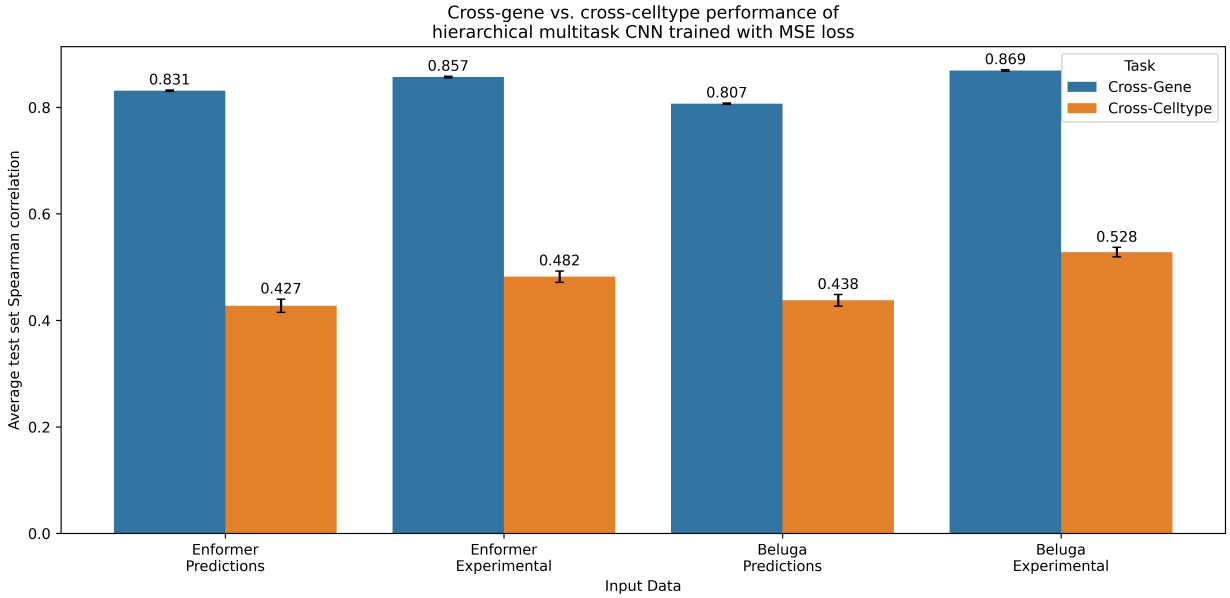

Figure 3.1: **Performance of hierarchical multitask CNNs trained with MSE loss on cross-gene and cross-celltype tasks.** For the cross-gene task (blue bars), we compute cross-gene test set Spearman correlation and report averages across all 218 cell types. For the cross-celltype task (orange bars), we compute cross-celltype test set Spearman correlation and report averages across all genes in the test set. We evaluate for hierarchical CNNs trained on four types of input data outlined on the x-axis. Error bars reflect ±1 standard error of the mean.

Results

We find that hierarchical models struggle on the cross-celltype task compared to the cross-gene task, as seen in Figure 3.1. For instance, while a hierarchical model that takes in Enformer predictions achieves a 0.831 cross-gene Spearman correlation, it only achieves a 0.427 cross-celltype Spearman correlation. We hypothesize the hierarchical strategy's poor cross-celltype performance may be due to the inherent difficulty of the cross-celltype task. The average cross-celltype coefficient of variation is 3.33 while the average cross-gene coefficient of variation is 7.79. In other words, on average, the expression values for a single gene across all tissues are closer together than the expression values for a single tissue across all genes.

The hierarchical models are trained with a Mean Squared Error (MSE) loss that penalizes differences between each predicted expression value (log RPKM) and its ground-truth measured expression value. Because the cross-celltype expression values are closer together, a model that is just trained with MSE loss will output expression values that may be quite close to the ground-truth measured expression values, but the ordering of the predicted ex-

pression values across cell types may be wildly incorrect. For instance, imagine a model that, for a given gene, just predicts the mean expression across cell types. The MSE loss of this model's predictions against the outputs is not too high because there is not much variation in expression values across cell types. However, this model does not make any meaningful celltype-specific predictions and performs poorly on the cross-celltype task. In summary, with MSE loss, it is more challenging to learn the minute differences between tissues. We need a loss that explicitly helps the model pay attention to cross-celltype differences in expression so that the model can optimize its predictions for the cross-celltype task.

Cross-celltype loss

To tackle this challenge, we introduce a hybrid loss function consisting of a cross-celltype Pearson correlation, designed to guide the model towards understanding in which tissues a gene is more highly expressed in than other tissues, and Mean Squared Error.

$$L(y, \hat{y}) = \alpha \cdot (1 - \text{Pearson}(y, \hat{y})) + \beta \cdot \text{MSE}(y, \hat{y})$$

α represents the weighting of the Pearson loss, and β represents the weighting of the MSE loss. We chose Pearson correlation instead of Spearman correlation for the loss function as it is faster to differentiate Pearson correlation than the differentiable version of Spearman correlation.

Methodology

Like the previous section, we train hierarchical multitask CNNs on top of four different types of input data: Enformer Predictions, Enformer Experimental, Beluga Predictions, and Beluga Experimental. Again, for Enformer input data, we only consider non-CAGE tracks.

We train each model with different hybrid loss functions. Each hybrid loss function has a different pair of weightings for the Pearson component and the MSE component. We additionally test hybrid loss functions that are equivalent to just MSE loss or just Pearson loss (i.e., one of the components has a weight of 0). Like before, we perform grid search to optimize hyperparameters for validation set performance, and we evaluate the cross-gene and cross-celltype performance on the test set.

Results

As the weighting of Pearson loss increases, we observe a slightly decay in performance on the cross-gene task, seen in Figure 3.2. We believe this is because as the weighting of Pearson loss increases, the influence of MSE loss on the direction of parameter updates drops. MSE loss helps the model consider the absolute magnitudes of the expression values. Compared to Pearson loss, which helps the model consider only the relative magnitudes of the expression values for a singular gene, MSE loss is more useful for helping the model rank expression

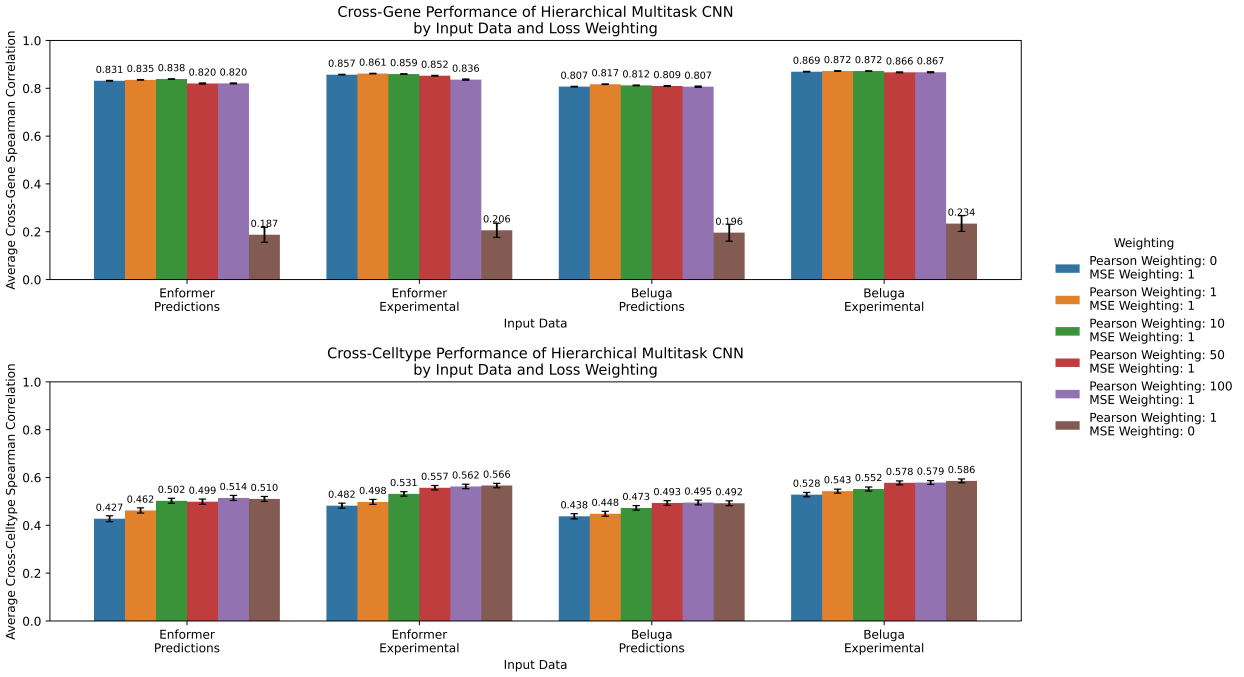

Figure 3.2: **Performance of the hierarchical multitask CNN trained with a hybrid loss function on cross-gene and cross-celltype tasks.** For the cross-gene task (top), we compute cross-gene test set Spearman correlation and report averages across all 218 cell types. For the cross-celltype task (bottom), we compute cross-celltype test set Spearman correlation and report averages across all genes in the test set (420 for Enformer, 495 for Beluga). We evaluate for 4 different types of input data, listed on the x-axis, and for 6 different hybrid loss functions, denoted by the different colored bars. Error bars reflect ± 1 standard error of the mean.

across genes. Indeed, when MSE loss is not included at all (i.e., weighted at 0), cross-gene performance significantly drops from around 0.8 to around 0.2 Spearman correlation for all four types of input data.

On the other hand, as the weighting of Pearson loss increases, we observe a significant jump in performance on the cross-celltype task, as seen in Figure 3.2. For example, when using Enformer predictions for regulatory chromatin features as the input to the hierarchical model, the average cross-celltype Spearman correlation jumps from 0.427 when weighting Pearson loss at 0 (blue bar in Figure 3.2) to 0.514 when weighting Pearson loss at 100 (purple bar), a 20.4% improvement. The jump occurs when using experimental chromatin features as input as well. When using Enformer experimental data for regulatory chromatin features as input, the average cross-celltype Spearman correlation jumps from 0.482 when weighting Pearson loss at 0 (blue bar) to 0.562 when weighting Pearson loss at 100 (purple bar), a 16.6% improvement.

3.2 Finetuning Enformer with a cross-celltype loss

Comparing hierarchical models and finetuning Enformer trained with cross-celltype loss

We used the hybrid cross-celltype loss function to finetune Enformer, and we compared finetuning Enformer with hierarchical modeling based on their performance on the cross-gene and cross-celltype tasks.

Methodology

We follow a similar methodology to finetune Enformer as described previously. Now, we finetune with the hybrid loss function, and we try 4 different hybrid loss function weightings.

Results

On one hand, hierarchical modeling and finetuning Enformer appear to perform comparably on the cross-gene task, according to Figure 3.3.

On the other hand, finetuning Enformer appears to outperform hierarchical modeling on the cross-celltype task, particularly when the Pearson loss is weighted low. Compared to hierarchical modeling, finetuning Enformer allows us to better take advantage of the more informative embedding space to achieve superior cross-celltype performance. Notably even an Enformer finetuned without any Pearson loss component outperforms a hierarchical model trained with only Pearson loss on the cross-celltype task.

However, as the weighting of the Pearson loss increases, the gap between cross-celltype performance hierarchical modeling and finetuning Enformer begins to close. When using the cross-celltype Pearson loss, the performance of the hierarchical modeling strategy can get quite close to the performance of finetuning Enformer. We note that hierarchical modeling can be more computationally efficient than finetuning Enformer, so it still remains a promising strategy.

Comparing hierarchical models and finetuning Enformer with linear probing baselines

We evaluated the performance of the linear probing baselines built in Chapter 1 on the cross-celltype task and compared this performance to the hierarchical CNNs and finetuned Enformer trained with 4 different hybrid loss functions.

Results

We find that when using hybrid loss functions with a lower weighting for the cross-celltype Pearson loss, hierarchical CNNs underperform the linear probing baselines on the cross-celltype task. As we increase the weighting of the cross-celltype Pearson loss, however,

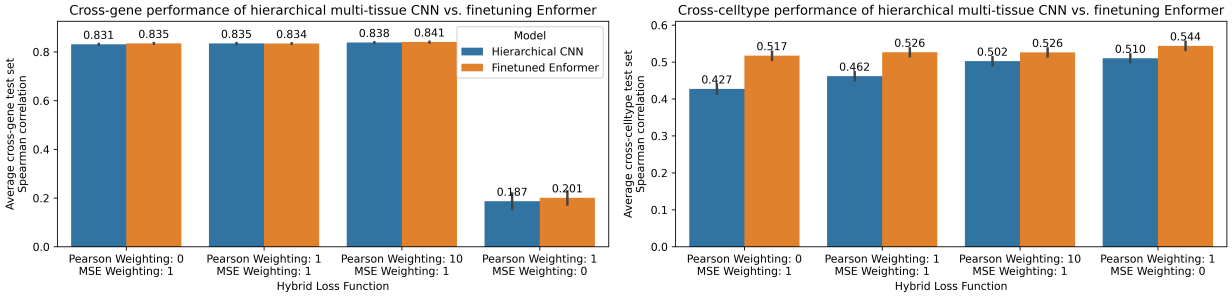

Figure 3.3: **Performance of hierarchical multitask CNN vs. finetuned Enformer, trained with the hybrid cross-celltype loss, on cross-gene and cross-celltype tasks.** We train hierarchical multitask CNNs (blue) and finetune Enformer models (orange) using 4 different hybrid loss functions (labeled on x-axis). For the cross-gene task (left), we compute cross-gene test set Spearman correlation and report averages across all 218 cell types. For the cross-celltype task (right), we compute cross-celltype test set Spearman correlation and report averages across all 420 genes in the test set. Error bars reflect ±1 standard error of the mean.

hierarchical CNNs do come to perform as well as or better than the baselines, demonstrating the effectiveness of the cross-celltype loss in helping the hierarchical model extract cross-celltype signal from the data. In contrast, we find that finetuned Enformer performs as well as or better than the linear probing baselines, for all hybrid loss functions tested even if the cross-celltype Pearson component is weighted at 0.

Surprisingly, we find that linear probes on the output tracks subset to just the CAGE tracks outperform linear probes on all the output tracks. We had expected the linear probes on all output tracks to outperform the CAGE-only probes because the linear probes on all output tracks get more data. We hypothesize that this surprising result may be due to the fact that the CAGE tracks contain more information about cell-type specificity. The non-CAGE tracks (i.e., the regulatory chromatin tracks like TF binding, histone modifications, and accessibility) may contain more noisy information about cell-type specificity, hurting the predictions when we include non-CAGE output tracks in our input.

Effect of freezing weights when finetuning Enformer

Finetuning Enformer is computationally expensive due to the large size of the Enformer model. Freezing some weights of Enformer when finetuning may reduce the computational cost of the backward pass (gradient calculation and parameter update). We investigate how freezing weights in Enformer when finetuning affects performance on both the cross-gene and cross-celltype tasks.

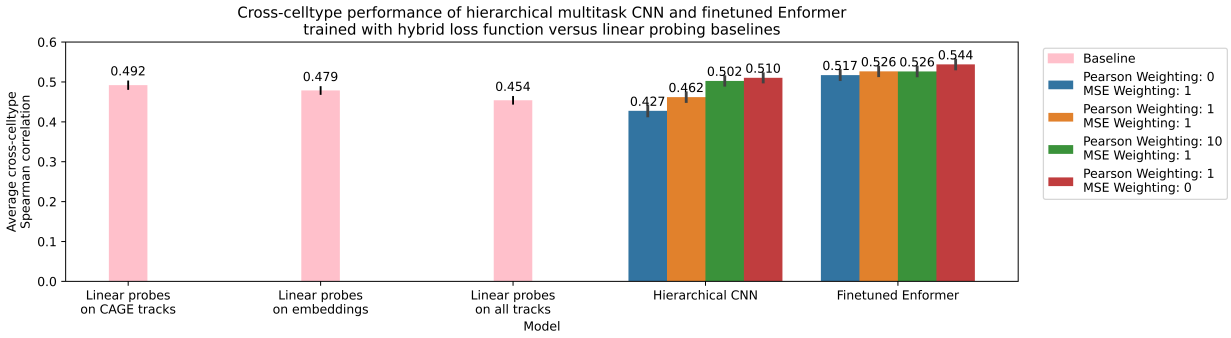

Figure 3.4: **Performance of linear probing baselines vs. hierarchical multitask CNN and finetuned Enformer trained with hybrid loss function on cross-celltype task.** Blue, orange, green, and red bars represent performance for hierarchical CNN and finetuned Enformer trained with four different hybrid loss functions. Pink bars represent performance for the linear probing baselines, which are not trained with a hybrid loss function. For each of the 420 genes in the test set, we compute the cross-celltype Spearman correlation across 218 cell types. We plot the average Spearman correlation across the 420 genes, with error bars reflecting ± 1 standard error of the mean.

Methodology

We follow the same methodology to finetune Enformer as described previously. When finetuning Enformer, we either leave all weights unfrozen as done previously or freeze the weights in the CNN and Transformer modules. This leaves only the parameters associated with the attention pool and final linear layer as learnable parameters during our finetuning process.

Results

Across all tested hybrid loss functions, we find comparable performance on both the cross-gene and cross-celltype tasks between finetuning Enformer with all weights unfrozen vs. CNN and Transformer modules frozen. Freezing the CNN and Transformer modules can be a way to reduce the computational cost of finetuning Enformer on our tasks and our data—and perhaps other tasks or datasets—while not sacrificing performance.

When we train with a nonzero cross-celltype Pearson loss component, we still find that unfreezing the CNN and Transformer modules does not improve performance. Differences in expression between cell types are known to often be biologically modulated by distal enhancers [3]. Current state-of-the-art models like Enformer often have trouble picking up on signal from distal enhancers [9]. It has been hypothesized that the use of a hybrid cross-celltype loss would help a model better pick up the signal from these distal enhancers. In Enformer, this would happen via parameter updates to attention layers in the Transformer module since these layers are responsible for integrating signals from distal elements [2].

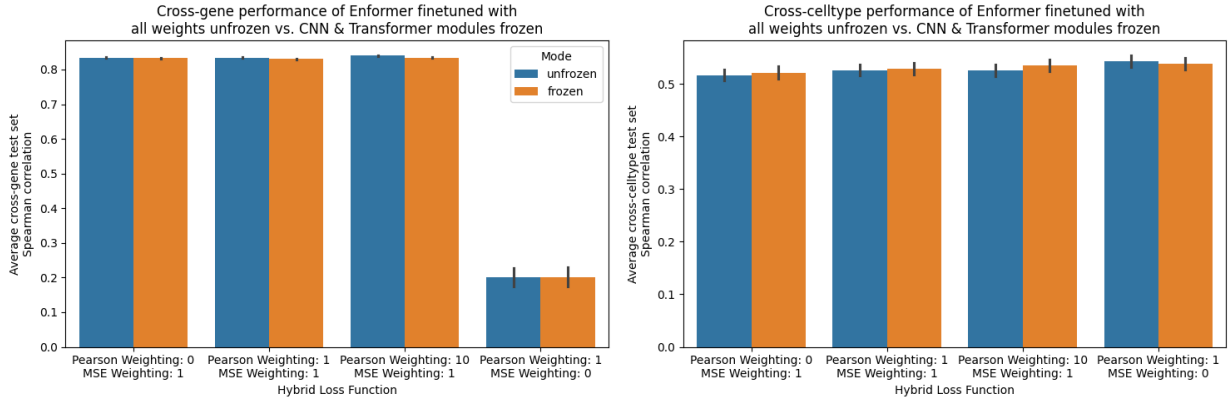

Figure 3.5: **Performance of finetuning Enformer with keeping all weights unfrozen vs. freezing CNN and Transformer modules on cross-gene and cross-celltype tasks.** Keeping all weights unfrozen is represented by blue bars ("unfrozen") vs. freezing CNN and Transformer modules is represented by orange bars ("frozen"). We finetune Enformer models using 4 different hybrid loss functions (labeled on x-axis). For the cross-gene task (left), we compute cross-gene test set Spearman correlation and report averages across all 218 cell types. For the cross-celltype task (right), we compute cross-celltype test set Spearman correlation and report averages across all 420 genes in the test set. Error bars reflect ± 1 standard error of the mean.

Therefore, we expect the effectiveness of a hybrid cross-celltype loss to depend on the Transformer module. However, what we find is that the effectiveness of a hybrid cross-cell-type loss is independent of whether we can update the parameters in the Transformer module. Thus, we hypothesize that the use of a hybrid cross-celltype loss does not help a model better capture the signal from distal enhancers.

Effect of cross-celltype loss on attention weights

To verify our hypothesis, we investigated whether finetuning Enformer with a hybrid cross-tissue loss would lead to changes in which bins the model pays attention to. We compare the following models: original Enformer, finetuning Enformer with just an MSE loss, and finetuning Enformer with a hybrid cross-tissue loss (Pearson loss weighting of 10, MSE loss weighting of 1). We pass in a 50kb sequence centered at the TSS for each gene in our test set into the model, and we average the attention weights across all layers and across all heads. In this experiment, we do not freeze any parameters when finetuning.

Attention maps

We consider all attention weights involving the TSS bin as the query and all other bins as the keys to compute an "attention map" for that gene. Averaging the attention maps over all the genes in our test set gives us Figure 3.6.

Our finetuning strategy for both MSE loss and the hybrid cross-celltype loss involves finetuning on only gene expression data (in particular, our dataset consists of 218 celltype-specific RNA-seq expression measurements per gene). Meanwhile, the original Enformer was trained on both gene expression data and regulatory chromatin data. Since distal enhancers play a critical role in regulating tissue-specific gene expression [7], finetuning on only gene expression data should make the model pay more attention to distal bins where enhancers may lie. Indeed, we do see that our finetuned models have higher higher attention weights than the non-finetuned Enformer model at the edges of the 50 kb sequence we test. Additionally, the difference in attention weights confirms that the process of finetuning does indeed update the weights in the attention layers.

However, the loss function with which we choose to finetune does not change how much attention the model pays to distal bins. In Figure 3.6, we find that the finetuned Enformer with MSE Loss and the finetuned Enformer with hybrid cross-celltype loss have nearly identical attention maps, especially in the distal regions. The hybrid cross-celltype loss is not getting the attention layers to pay special attention to the distal enhancers that capture much of the cross-celltype differences [3].

Distal bins

To support this point, we additionally calculate the number of high-attention distal bins to quantitatively determine if there is a difference between the base Enformer, the finetuned Enformer with MSE loss, and the finetuned Enformer with a hybrid cross-celltype loss in terms of attending to distal bins. If the cross-celltype loss supposedly helps models pay more attention to distal enhancers, we would expect to see an increase in the number of high-attention distal bins for finetuned Enformer with a hybrid cross-celltype loss. We define distal bin as over 50 bins (i.e., greater than 6.4 kb) away from the TSS. We calculate the number of high-attention distal bins for several attention thresholds and plot our results in Figure 3.7.

Finetuning Enformer (orange and green lines) appears to increase the number of high-attention distal bins compared to the original Enformer model (blue line). We again see that finetuning on gene expression data increases the attention paid to distal regions of the sequence since distal information is important for gene expression prediction.

However, between the two finetuned models, there does not appear to be a significant difference in high-attention distal bins. Compared to the MSE loss function, the use of a hybrid cross-celltype loss function does not seem to increase the attention paid to distal bins where distal enhancers responsible for celltype-specific expression may lie.

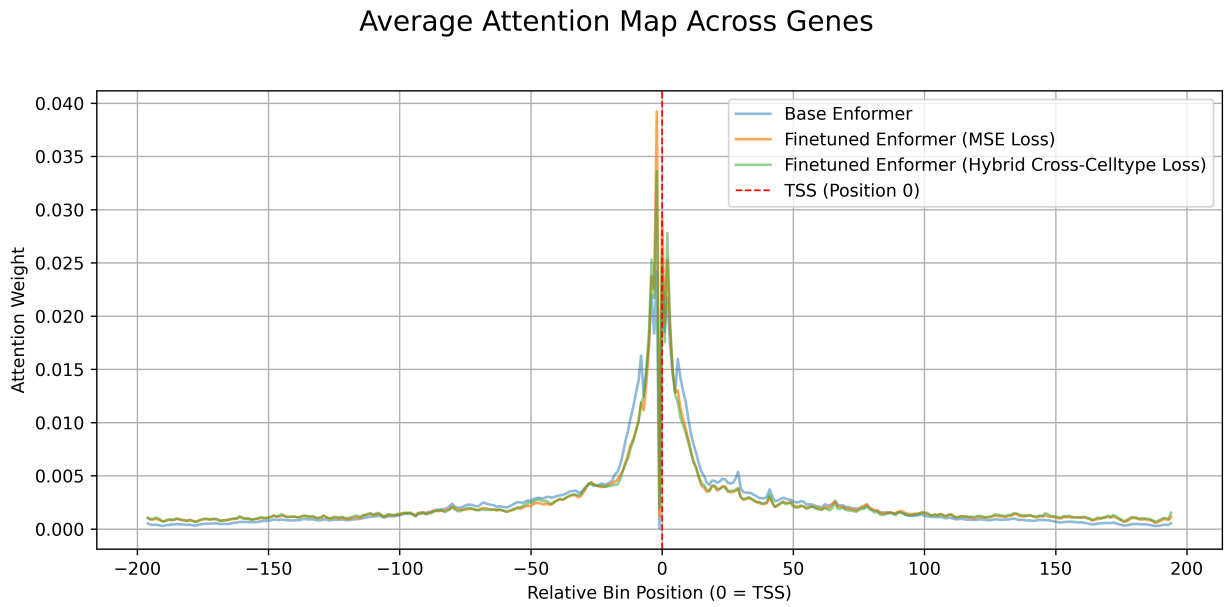

Figure 3.6: **Map of attention weights for different Enformer models.** We analyze base Enformer model (blue), Enformer finetuned with MSE loss (orange), and Enformer finetuned with hybrid cross-celltype loss (green). The x-axis lists the bin position relative to the TSS. The y-axis represents the average attention weight, computed by averaging all the attention weights with a query of 0 (i.e., center bin, or TSS) over all layers, heads, and genes. The vertical red dashed line indicates the TSS at bin 0.

3.3 Conclusion

In this chapter, we begin by showing that our hierarchical modeling approach does not perform well on the cross-celltype task compared to the cross-gene task. We propose a hybrid loss function that consists of a cross-celltype Pearson component and show that it is effective in improving performance of our hierarchical models on the cross-celltype task with minimal change in performance on the cross-gene task. We additionally look at finetuning Enformer with the cross-celltype task and find that finetuning Enformer offers better performance on the cross-celltype task than hierarchical modeling. We observe that the more computationally-efficient strategy of freezing the CNN and Transformer modules of Enformer while finetuning does not impact performance on the cross-gene or cross-celltype task. Finally, we discuss how finetuning Enformer with a cross-celltype loss does not help Enformer pay more attention to distal information, where enhancers responsible for tissue-specific expression may lie. Future work can look at other cross-celltype loss functions and their effectiveness both for training hierarchical models and for finetuning Enformer. In particular, we leave it to future work to find a loss function that can help Enformer generate better tissue-specific expression predictions by paying more attention to distal enhancers.

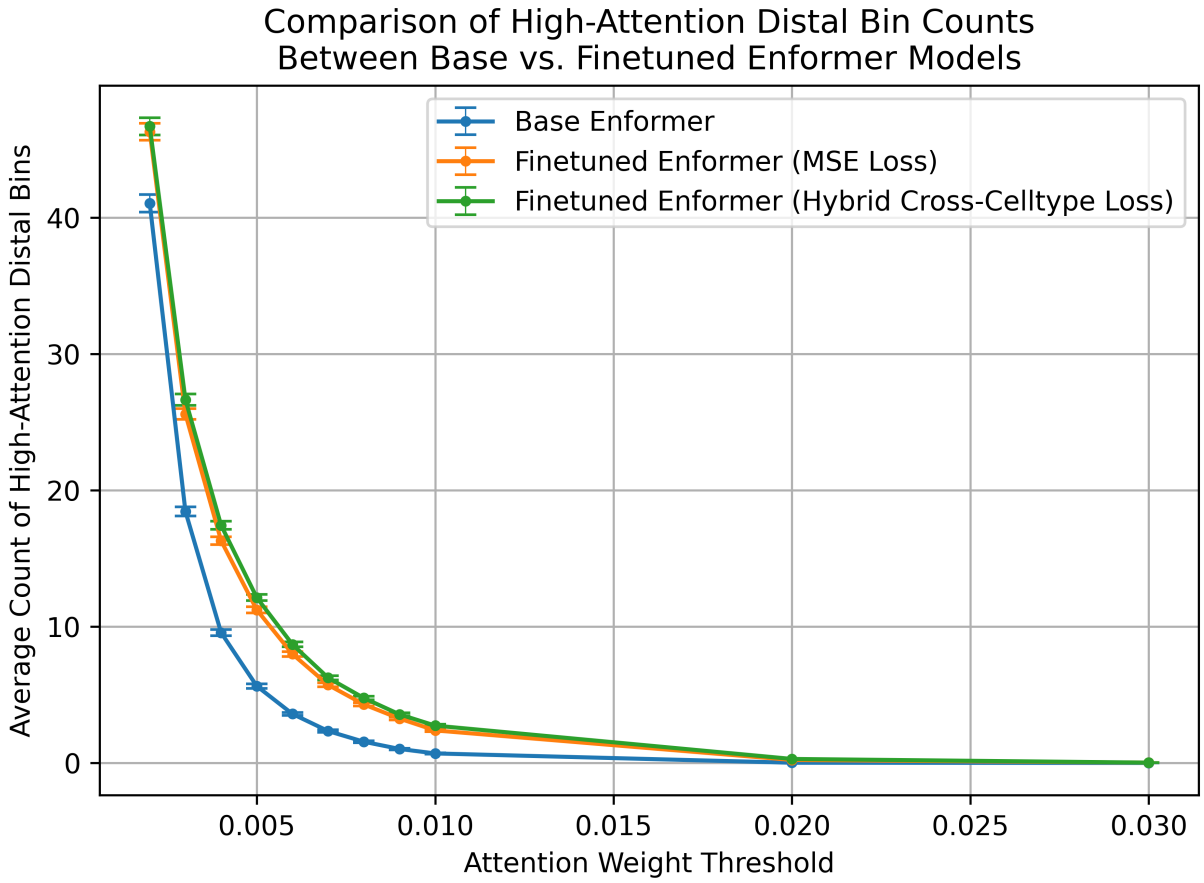

Figure 3.7: **Average counts of high-attention distal bins for three different Enformer models.** The three Enformer models are the base Enformer model, the Enformer model finetuned with MSE loss, and the Enformer model finetuned with a hybrid cross-celltype loss. We test out several attention thresholds from 0.002 to 0.03. For each threshold, we calculate the number of distal bins (defined as greater than 6.4 kb away from TSS) where the average attention weight across all layers and all heads exceeds that threshold. We plot the average number of these high-attention distal bins across all genes. Error bars reflect ± 1 standard error of the mean.

Bibliography

[1] "A promoter-level mammalian expression atlas". In: *Nature* 507.7493 (2014), pp. 462–470.

[2] Žiga Avsec et al. "Effective gene expression prediction from sequence by integrating long-range interactions". In: *Nature methods* 18.10 (2021), pp. 1196–1203.

[3] Michael Bulger and Mark Groudine. "Functional and mechanistic diversity of distal transcription enhancers". In: *Cell* 144.3 (2011), pp. 327–339.

[4] ENCODE Project Consortium et al. "An integrated encyclopedia of DNA elements in the human genome". In: *Nature* 489.7414 (2012), p. 57.

[5] Francis H Crick. "On protein synthesis". In: *Symp Soc Exp Biol.* Vol. 12. 138-63. 1958, p. 8.

[6] Ran Elkon and Reuven Agami. "Characterization of noncoding regulatory DNA in the human genome". In: *Nature biotechnology* 35.8 (2017), pp. 732–746.

[7] Sven Heinz et al. "The selection and function of cell type-specific enhancers". In: *Nature reviews Molecular cell biology* 16.3 (2015), pp. 144–154.

[8] François Jacob and Jacques Monod. "Genetic regulatory mechanisms in the synthesis of proteins". In: *Journal of molecular biology* 3.3 (1961), pp. 318–356.

[9] Alexander Karollus, Thomas Mauermeier, and Julien Gagneur. "Current sequence-based models capture gene expression determinants in promoters but mostly ignore distal enhancers". In: *Genome Biology* 24.1 (2023), p. 56.

[10] David R Kelley. "Cross-species regulatory sequence activity prediction". In: *PLoS computational biology* 16.7 (2020), e1008050.

[11] David R Kelley, Jasper Snoek, and John L Rinn. "Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks". In: *Genome research* 26.7 (2016), pp. 990–999.

[12] David R Kelley et al. "Sequential regulatory activity prediction across chromosomes with convolutional neural networks". In: *Genome research* 28.5 (2018), pp. 739–750.

[13] Tony Kouzarides. "Chromatin modifications and their function". In: *Cell* 128.4 (2007), pp. 693–705.

[14] Roadmap Epigenomics Consortium Integrative analysis coordination Kundaje Anshul 1 2 3 Meuleman Wouter 1 2 Ernst Jason 1 2 4 Bilenky Misha 5, Scientific program management Chadwick Lisa H. 53, and Principal investigators Bernstein Bradley E. 2 26 42 Costello Joseph F. 14 Ecker Joseph R. 9 Hirst Martin 5 18 Meissner Alexander 2 6 Milosavljevic Aleksandar 7 Ren Bing 8 13 Stamatoyannopoulos John A. 10 Wang Ting 21 Kellis Manolis 1 2. "Integrative analysis of 111 reference human epigenomes". In: *Nature* 518.7539 (2015), pp. 317–330.

[15] Johannes Linder et al. "Predicting RNA-seq coverage from DNA sequence as a unifying model of gene regulation". In: *Nature Genetics* (2025), pp. 1–13.

[16] John Lonsdale et al. "The genotype-tissue expression (GTEx) project". In: *Nature genetics* 45.6 (2013), pp. 580–585.

[17] Jian Zhou and Olga G Troyanskaya. "Predicting effects of noncoding variants with deep learning–based sequence model". In: *Nature methods* 12.10 (2015), pp. 931–934.

[18] Jian Zhou et al. "Deep learning sequence-based ab initio prediction of variant effects on expression and disease risk". In: *Nature genetics* 50.8 (2018), pp. 1171–1179.