CS373 Final Project Report

Anvita Gupta

June 7, 2016

1 Introduction

Chromatin accessibility for transcription and translation is a key measure of gene regulation in cells, providing insight into the epigenome of a cell and how the epigenome changes over time. In turn, the differences in cells epigenomes lead to cellular specialization for different functions, and epigenetic changes have been linked with aging and the progression of different diseases like Cancer and Alzheimer's [].

Chromatin accessibility assays seek to measure how tightly the chromatin is bound to the nucleosomes that hold it together. The less tightly the chromatin is bound, the more accessible it is to regulatory elements. In the past, chromatin accessibility could be inferred from ChIP-seq assays, which measured the ability of histone proteins to bind to different regions in the DNA. However, chromatin accessibility assays like MNase-Seq and ATAC-Seq have been developed to directly measure the effect of chromatin accessibility changes on gene expression. However, these assays have largely only been applied to isolated cells *in vitro*.

Recently, the Brunet lab at Stanford has created the first high resolution ATAC-Seq dataset to measure whole genome chromatin accessibility changes from embryo to adulthood in the model organism *C. Elegans*. *C. Elegans* is a highly studied model organism; this dataset records chromatin accessibility at three stages of its development: early embryo (EE), larval stage 3 (L3), and young adult (YA). In addition to accessibility at each of these three stages, the dataset also contains information about the changes in accessibility between each stage.

One key question which this dataset enables us to gain more infomation about is what regulatory elements are most active in each stage of the development of *C. Elegans*, and how these regulatory elements change over time. This is the question that this project aims to answer, using a machine learning approach.

Identifying novel regulatory elements in the *C. elegans genome* has several important implications. Firstly, since *C. Elegans* is a well studied model organism, it serves as a proof of concept to show that our machine learning approach applied to this ATAC-Seq data can identify already known regulatory elements in the *C. Elegans* genome. Secondly, regulatory elements are often well conserved, and so regulatory elements in *C. Elegans* may be conserved in other organisms. Finally, identifying how important regulatory elements change over time gives us key insight into the epigenetics of embryonic development and aging.

We first train and tune tune several different deep learning models to accurately predict which regions of the genome are differentially expressed in the stages of worm maturation. These models predict which regions of the *C. Elegans* genome are differentially expressed in each stage of development, using the genetic sequences of the regions as features. After picking the best model, we rank features using DeepLift in order to analyze which motifs are most important in predicting whether a region is differentially expressed between two developmental stages.

1.1 Previous Research

Convolutional Neural Networks and deep learning have rapidly emerged as a state of the art tool for learning important DNA motifs in functional genomics. CNNs have previously been implemented by Kelley et. al to predict DNA accessible regions in 164 different cell types and their DNA-Seq chromatin accesibility data Kelley et al. [2015]. The resulting trained CNNs have been released into the open source as a package known as Bassett. Bassett predicts change in accessibility between two variant alleles, finding that genes with SNPs with a larger change in accessibility were more likely to play a causal role for the phenotype observed.

In addition, the Bassett architecture uses in silico mutagenesis to find the specific nucleotides driving accessibility. Essentially, in silico mutagenesis is another form of feature ranking, where each nucleotide in the DNA region is mutated, and the change in predicted accessibility is assessed. Intuitively, nucleotides that play an important role in accessibility (nucleotides that are protein binding motifs, for example) would lead to the greatest change in predicted assessibility when modified. One drawback to the approach of feature ranking using in silico mutagenesis, however, is that all possible DNA mutations of all possible regions must be computed, which requires a great deal of computational power. Gradient based methods have commonly been used to rank important input features into a neural network, where the importance of a feature is correlated with the gradient in the neuron that reads that feature. However, these gradient based methods fail to take into account that in activation functions such as the ReLu (rectified linear unit) activation functions, the gradient will be 0 if the input is negative, although this input may have great importance for predicting the final output. Thus, we use the deeplift feature ranking method Shrikumar et al. [2016] to find which motifs are most important for predicting whether a particular region of DNA is accessible at some time point.

2 Methods

2.1 Dataset

The ATAC-seq dataset consists of stage specific peaks, and sets of regions that were significantly differentially expressed between different stages. As previously mentioned, there are three different stages: EE, L3, and YA. Each peak in the dataset is associated with one of the five chromosomes of *C. Elegans*, a starting position on the chromosome, and an ending position. In total, there are over 55,000 peaks in the three regions. Peaks which significantly changed accessibility between two stages are recorded in the differentialPeaks expression file, along with whether they were upregulated or downregulated from the first stage to the second stage. The differential peaks are recorded from EE to L3, L3 to YA, and EE to YA. In total, there are about 22,200 differentially expressed peaks, where each peak is identified by its start position, end position, and chromosome.

In order to prepare the dataset for input into our deep learning model, we first merge all overlapping intervals from the stage specific and differential peaks. We then create 200bp sliding windows with a stride of 100 from these peaks, making sure that every base pair in a peak is included in some window. We then intersect each window with the peaks from each of the three stages and with the differentially expressed regions. If some window overlaps more than 50% with some stage, we record that the window is expressed in that stage. Note that in addition to the three windows for each stage, each differentially expressed region (ex. $EE - L3$) has 3 windows of its own for up, down, and no change between the early embryonic and larval 3 stages.

For example, consider some window $[100, 300]$. If this window overlaps by 150bp with EE and 100bp with L3, as well as 180bp with $EE - L3 - upregulated$, then we would record 1 for this window in these corresponding columns, and 0 in all the other columns.

This file constitutes the labels file for each region. The model itself will be trained to predict these labels using the genetic sequence for each region, plus a 400 bp flank to each side of the region. Thus, each ”datapoint” is an input of 1000 basepairs, followed by 3 predictions for the stage specific regions and 9 predictions for each of the three transitions (3 per transition).

We set aside roughly 20% of the data (from Chromosome III) to be our test set, and we set aside another 20% of the data (from Chromosome IV) to serve as our validation set. Chromosomes I,II,IV, and V were used as training data for the model. By putting only sequences from one chromosome in the test set and validation set, we ensure that no loci overlap between our training and test set, which would bias the performance of the model.

2.2 Deep Learning Model

Several CNNs with different architectures were trained with the Keras framework and Theano on one-hot encoded sequences of the 1000 bp regions for each peak in the training set.

Each CNN had six different tasks: the first task was to make three binary predictions for whether a peak was present in the EE,L3, and YA stages. The next set of three tasks was to predict whether a genomic region was upregulated, downregulated, or had no change from EE to L3, EE to YA, and YA to L3. These tasks are multiclass classification tasks. In all of the architectures, a sigmoidal activation function was used for the binary tasks, and a softmax activation function was used for the multiclass tasks.

The first architecture we trained on the *C. Elegans* data contains three convolutional layers with a maxpooling layer after each convolutional layer, followed by two fully connected layers. Each neuron in the convolutional layers had a ReLU activation function. The first convolutional layer contains 300 filters of length 19 and stride 1, followed by a maxpooling layer with stride and width of 3. A second convolutional layer followed, with 200 filters of length 11 and a maxpooling layer with stride and width of 4. The third and final convolutional layer had 200 filters of length 7, and its maxpooling layer also had a stride and width of 4 base pairs.

Each neuron in the fully connected layer had a parametric rectified linear unit activation function (PReLU). The PReLU is a generalization of the traditional rectified unit (ReLU) and has been shown to improve model fitting without much greater computational time and with little risk of overfitting He et al. [2015]. Each fully connected layer contains 1000 neurons, with a dropout rate of 0.3.

Note that minibatch gradient descent with momentum updates was used to train the model. Our batch size was 200, so on each iteration, we train the network on 200 of the examples from the training set.

We ran this model on the *C. Elegans* data, once without correcting for class imbalance, and once correcting for it. We correct for class imbalance by weighting examples from underrepresented classes more highly. For example, if there were two classes a and b, with n_a examples from a and n_b examples from b, then we would give a the weight of $(n_a + n_b)/n_a$ and we would give class b the weight of $(n_a + n_b)/n_b$. Then the total weight assigned to predicting class a and b is the same, $(n_a + n_b)$. The class imbalance statistics for the *C. Elegans* dataset are shown for the binary tasks in Figure 1 and for the multiclass classification tasks in Figure 2.

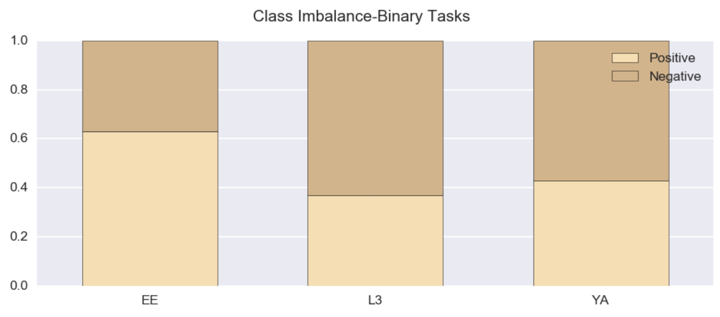


Figure 1: In this graph, if a datapoint is classified as positive for a given stage, it means that region of DNA was found as accessible during that stage. The class imbalance is not as great as for the differential classification tasks.

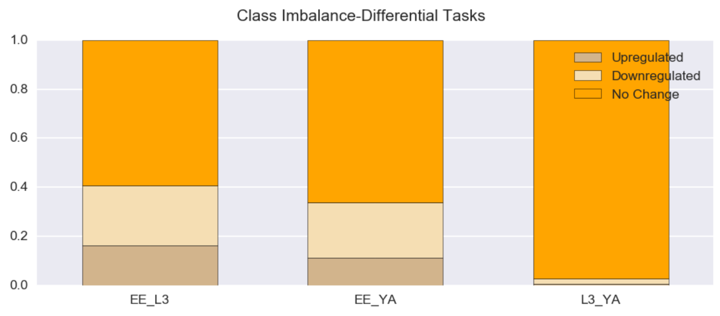


Figure 2: Upregulated datapoints are shown at the bottom of the graph, downregulated examples are stacked in the middle, and datapoints that did not change between the stages are shown in orange. The proportion of examples with no change is much greater from L3 to YA than from EE to L3 or EE to YA. This is as expected, because chromatin accessibility changes are correlated with development, and developmental changes occur more at the earlier stages of an organism's development.

The second architecture of CNN we trained was slightly more simplified than the first, although the basic architecture of the neural network remained the same in the number and order of fully connected, convolutional, and maxpooling layers. However, in this architecture the first convolutional layer had 250 filters with a length of 4, although its maxpool layer stride and width remained the same at 3. The second convolutional layer had 500 filters with a length of 4, but its maxpool layer had a larger stride and width of 7. The last convolutional layer again had 250 filters with a length of 4, and its maxpool layer had a width and stride of 7.

In the third and final architecture, the maxpooling layers were removed from every convolutional layer, and replaced with a single maxpooling layer following all of the convolutional layers. The parameters of the output layers, fully connected layers, and convolutional layers remained the same.

2.3 Model Tuning

With the third architecture, a random search was performed for the best hyperparameters. The first hyperparameter optimized was the maxpool layer stride and width size. The model's performance was tested with a stride and width of 10,20,40,70,and 100.

2.4 Feature Importance using Deeplift

We took the best performing model (the one with the second architecture) and run the Deeplift feature ranking framework on the true positives from the binary tasks. Both multiplier scores and Deeplift scores are calculated for the true positives in our dataset. We use the true positives found at the 50% threshold for each of tne binary tasks. At this threshold, there were 3844 true positives in the Y3 stage, 4118 true positives in the L3 stage, and 13114 true positives in the EE stage.

At a high level, Deeplift evaluates which parts of the sequence (called ”seqlets”) are most important in predicting whether a particular peak is predicted as positive during each stage. Deeplift assigns both deeplift scores and multiplier scores **?** to each seqlet, which is about 15 bp long. Using these multiplier scores, a cross correlation matrix is calculated for the seqlets. The seqlets that are most highly correlated are classified into the same cluster. For each cluster, we identify the parts of the sequence that the most seqlets have in common, and we display their position weight matrices, Deeplift scores, and multiplier scores. These motifs are what the model is learning to use to identify a particular data point as being called during a particular region.

We ran Deeplift to assign an importance score to each base pair in the true positive sequences. Regions with high predicted deep lift scores were clustered into seqlets of 15bp. The seqlets are then aligned with eachother, and a correlation matrix is computed for all the seqlets. We can plot the correlation matrix using a t-sne embedding to get a rough idea of the number of clusters. Based on this idea, we use k means clustering to divide the seqlets into different clusters. We then find the motifs which had the highest number of seqlets that aligned to them.

3 Results and Discussion

Note that for each architecture we have 6 classification tasks: three stage-specific binary classification tasks and three differential multitask classification tasks. The overall auROC reported for a model is the averaged auROC for all of these 6 tasks.

3.1 Architecture 1

The best validation auROC Architecture 1 was able to achieve was 0.7808; however, at the iteration with the best validation performance, the model achieved an accuracy of 0.853. In addition, the best performance of the model on the training set was 0.9315, which indicates that the model is overfitting on the worm data. As auROC is sometimes an inflated measure of performance (since auROC starts at 0.5), we also report the auPRC on the validation set, which is 0.492, only slightly better than the much less complex architecture 3. This provides further evidence that the model is overfitting to our training data.

We provide a breakdown of the auROC values for the multiclass differential tasks in Table 3.1. The auROC values for the binary tasks, as well as a comparison with these values for the other architectures, are shown in the comparison section.

Architecture 1	L3_YA	EE_YA	EE_L3
auROC on Positives	0.721	0.788	0.859
auROC on Negatives	0.739	0.853	0.805

Table 1: AuROC values for differential task predictions for Architecture 1. Note that since this was a multitask classification task, the auROCs were computed using a ”one vs rest” metric to get the true positive and the false positive rates.

3.2 Architecture 2

Architecture 2 performed best on the *C. Elegans* data set, reaching an auROC of 0.7964 on the validation set. However, this model still showed signs of overfitting, as the training performance on the epoch with the best validation performance was 0.8654, about seven percent higher than the model's performance on the validation set. However, the overfitting was still less than the overfitting with architecture 1. At the epoch with the best validation performance, the auROC on the test set was 0.803. The auROC can sometimes be an inflated measure of performance, as the auROC for a completely random model is only 0.5. Consequently, we report the auPRC as well, which is 0.512 for Architecture 2. Table 3.4 shows that this auPRC level is higher than the auPRC measures for any other architecture, further validating that Architecture 2 is the best one tested in terms of striking a balance

Architecture 2	L3_YA	EE_YA	EE_L3
auROC on Positives	0.785,	0.795,	0.863,
auROC on Negatives	0.742	0.861	0.815

Table 2: AuROC values for differential task predictions for Architecture 2. Note that since this was a multi-task classification task, the auROCs were computed using a "one vs rest" metric to get the true positive and the false positive rates. Just as expected, the task specific auROC values for architecture 2 are higher than those of architecture 1.

between overfitting and underfitting (though the model is still overfitting). The task specific auROCs for the differential tasks are shown in Table 3.2

We chose to use this particular architecture because Bassett implements a neural network with a similar structure and achieves good results Kelley et al. [2015]. This architecture also performed the best out of all three of the architectures we tested, and was our basis for later ranking important motifs using deeplift.

3.3 Architecture 3

We further simplified the architecture, as overfitting was still observed with Architecture 2. Out of all the maxpool strides and widths we tested (10,15,40,70,100), the model with maxpool stride of 40 performed best in terms of validation performance. Figure 3 shows the auROCs for the training, validation, and test set at each maxpool setting. Maximum overfitting is seen to occur at a maxpool stride and width of 70, and the model does not learn at all at a maxpool width of 15. This is not unusual; it means that the maxpool combination of 15 along with the rest of the parameters meant that the model was not able to learn at all from the dataset.

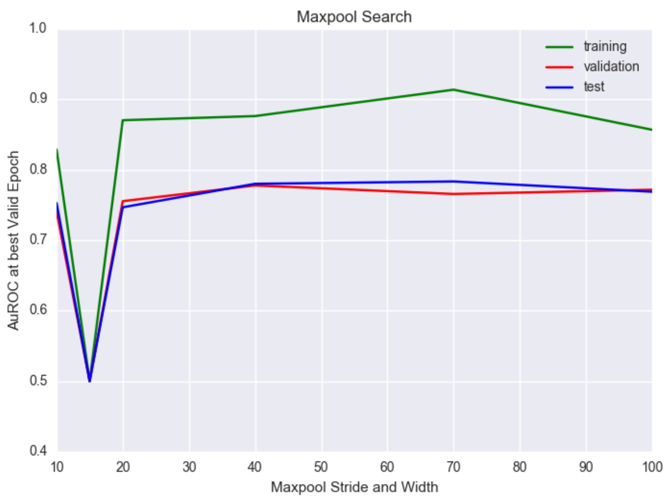


Figure 3: This image shows the training, test, and validation auROC values for each of the different maxpool settings. AuROC values were measured at the iteration with the best validation accuracy. The maxpool settings were 10,15,20,40,70, and 100. As can be seen from the graph, the maximum difference between the training and validation sets occurs when the maxpool stride is 70. The model fails to learn at all when the maxpool setting is 15, even upon multiple iterations of this run.

3.4 Comparison

Figure 4 shows the difference in validation auROCs for the binary tasks, which, again, were predicting 0/1 for whether a peak was accessible in a certain stage of development. As can be seen in the figure, the validation auROCs do not vary drastically for the three architectures; we would then expect that most of the difference in auROC for these architectures comes from the difference in auROC on the multiclass classification tasks.

As can be seen from close examinations of Table 3.2 and Table 3.1, most of the gain in performance by Architecture 2 comes in the prediction of positives in the L3_YA differential task. L3_YA positives were especially

Architecture 3	L3_YA	EE_YA	EE_L3
auROC on Positives	0.772,	0.774,	0.846,
auROC on Negatives	0.722	0.841	0.795

Table 3: AuROC values for differential task predictions for Architecture 3 with a maxpool stride of 40 (which performed the best out of all the maxpool strides). Note that since this was a multitask classification task, the auROCs were computed using a "one vs rest" metric to get the true positive and the false positive rates. Just as expected, the task specific auROC values for architecture 3 are lower than those of architecture 1 and 2, reflecting the ranking in the accuracies as well.

	auROC	auPRC	recallAtFDR20	balancedAccuracy	unbalanced Accuracy
Architecture 1	0.781	0.492	0.284	0.604	0.786
Architecture 2	0.796	0.512	0.348	0.612	0.815
Architecture 3	0.778	0.490	0.291	0.614	0.806

Table 4: Performance Metrics for Different Architectures

imbalanced (present in only 0.005249 proportion of the total number of L3_YA datapoints), and so they were weighted very highly. Thus, the improvement of Architecture 2 in predicting even a few data points in the L3_YA positives class would show up as a very large increase in auROC since those data points were given so much weight.

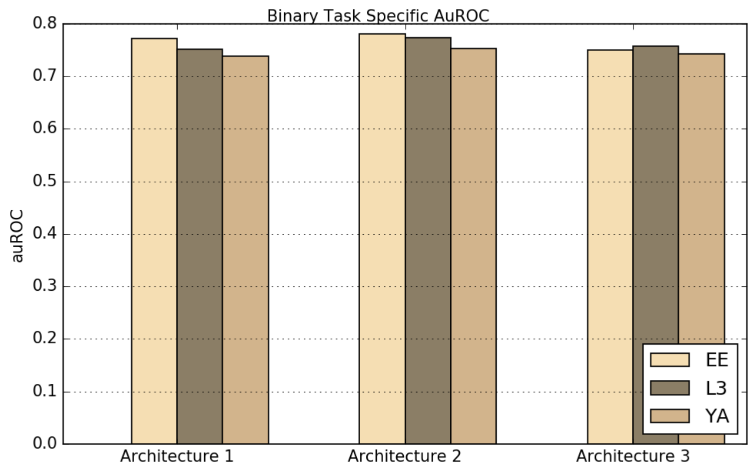


Figure 4: We compare the validation auROCs for the binary tasks of predicting a peak's accessibility during the EE, L3, and YA stages.

3.5 Motif Finding using Deeplift

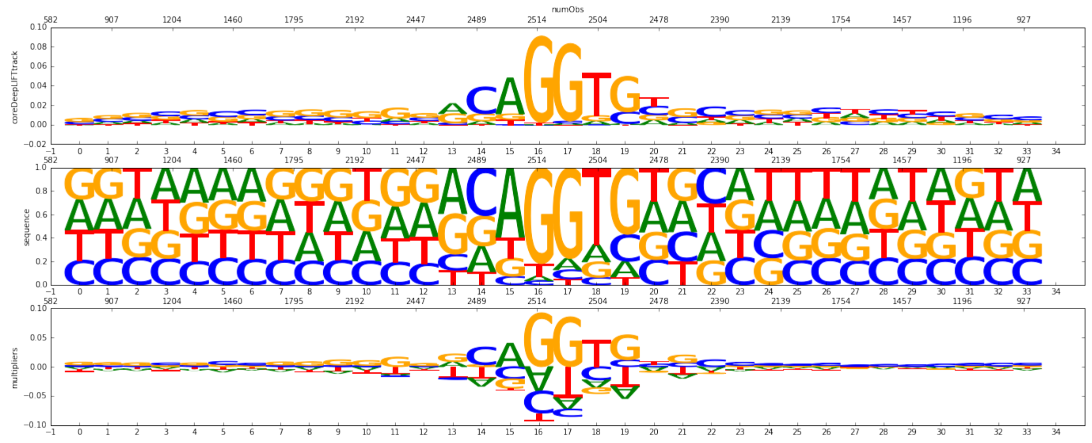
Figure 5, 6, and 7 show the two motifs that deeplift identified as most important for predicting whether a particular subsequence was accessible at a particular stage. The GAGA motif was found as one of the most important motifs for predicting presence of chromatin accessibility in all three of the developmental stages.

The GAGA motif is a well-studied motif that is conserved in organisms from plants to humans Harfe et al. [1998] and has been identified to play a role in chromatin structure dynamics and accessibility. This provides validation that the model is learning more than just GC content, as it is using this GAGA motif as one of the most important predictors of whether a particular region of DNA is accessible. The fact that the GAGA motif is one of the top two most important motifs in every stage of development might indicate that the GAGA motif is an important predictor of accessibility across all stages of development, and is thus not a differential predictor of whether a region of DNA is accessible in one stage versus another. One thing to note, however, is that in the L3 and YA stages the GAGA motif very clearly alternates between AGAGAG. However, in the EE stage, the GAGA motif is rather AGAGGG, where the Gs in later positions are assigned higher weight than the As, although both Gs and As are present at similar quantities in the seqlets.

We see this pattern in the GGCGCCG motifs of the L3 and YA stages as well, where the dominant motif is followed by different nucleotides at the flanks. For example, in the L3 stage the motif is GGCGCCG, where the G is present in almost 0.8 percent of all the seqlets. In the YA stage, however, the motif is GGCGCCT, and the G is percent in around 0.7 percent of all the seqlets, slightly less than in the L3 stage. This may indicate that certain motifs are important for accessibility of a region in general, but small differentiations of the motifs at the flanks are correlated with whether a particular region of DNA is differentially accessible in the L3 or the YA regions.

Besides the GAGA motif, the other motif found to be most important in the EE stage was completely different from the GGCGCCG motif found in the L3 and YA stages. This is consistent with our expectation that DNA accessibility changes occur most between the EE and L3 stages, and less so between the L3 and YA stages as the organism's developmental rate slows.

Motif 1:



Motif 2:

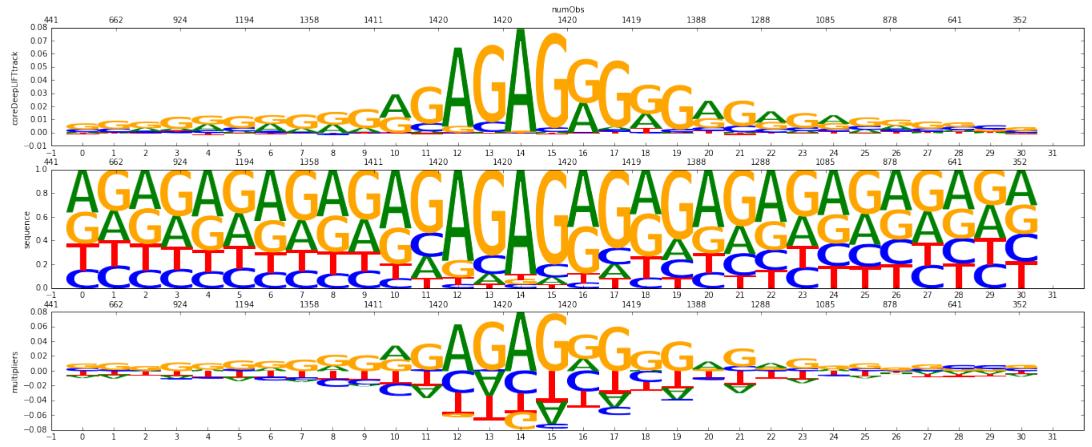
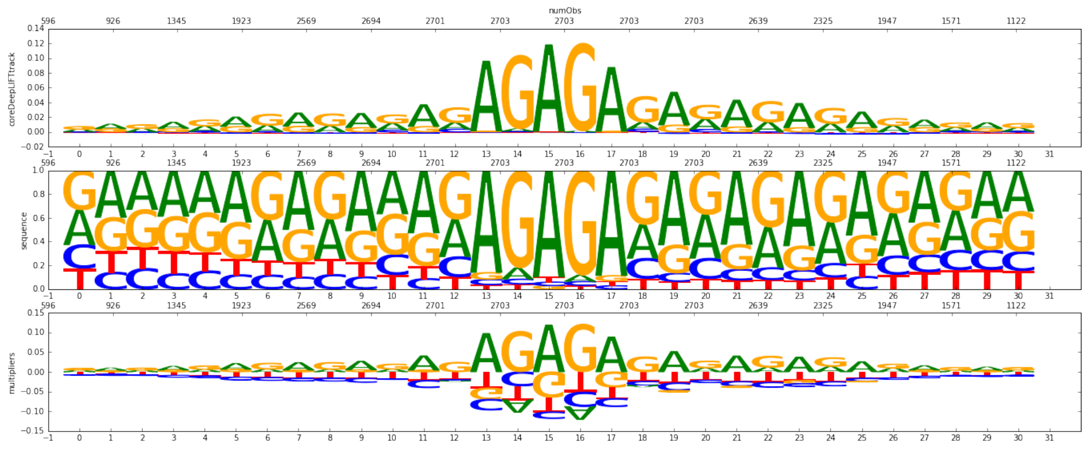


Figure 5: These two motifs were found to be most predictive of true positives in the Early Embryonic development stage, meaning that sequences with these motifs present were more likely to be predicted as positive (accessible) in the EE stage by the deep learning model. The first track shows the deeplift scores assigned to each base pair in the sequence, where the height of the base pair is proportional to its score, and thus its importance. The second track shows the position weight matrix, and the third track shows the scores computed by the multiplier method for each position. The main difference between the deeplift method of computing scores and the multiplier method is that the multiplier method also displays which nucleotides the model does not want to see at a particular position (e.g. which would cause the model to predict that particular sequence as a negative), whereas the Deeplift scores show only which nucleotides the model wants to see (e.g. which help the model give a positive position). One of the most important motifs found was the GAGA motif (Motif 2), which has been validated as a well-conserved motif connected with chromatin structure dynamics and accessibility changes.

4 Future Work

One possible limitation in the model we have developed currently is that only peaks whose height have changed significantly (p value = 0.05) will be called as "upregulated" or "downregulated" between different stages. A peak

Motif 1:



Motif 2:

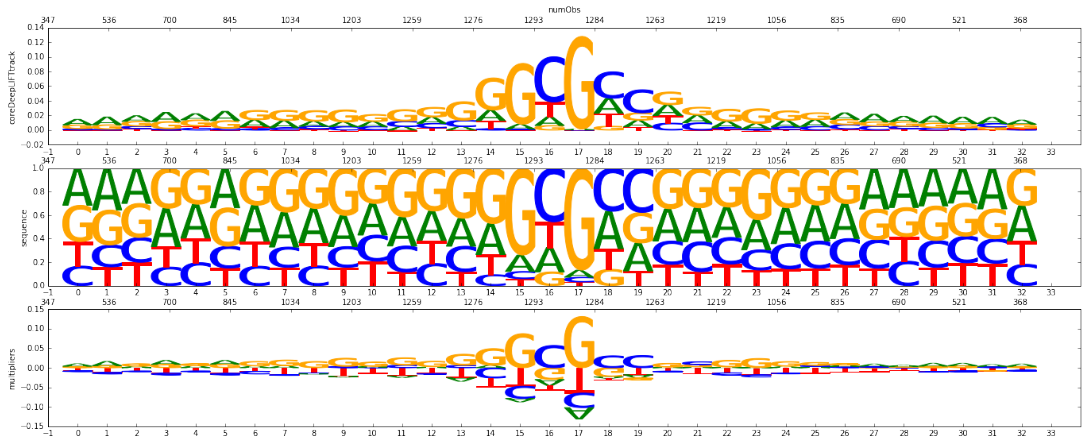
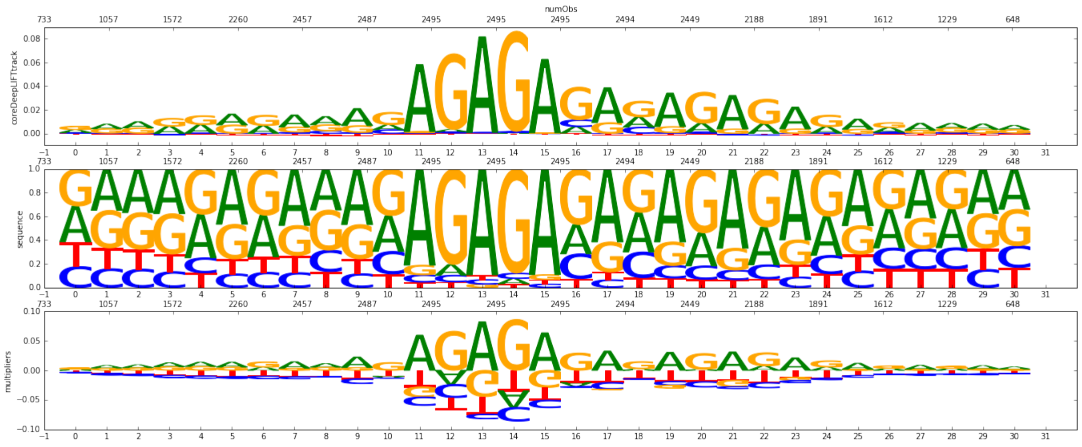


Figure 6: These two motifs were found to be most predictive of true positives in the L3 development stage. The descriptions of the three tracks in each motif are given in Figure 5. As in the EE stage, the GAGA motif was found to be highly correlated with DNA accessibility in the L3 stage as well. The second GGCGCCG motif resembles the second motif in the Y3 stage, with the difference of the final G.

Motif 1:



Motif 2:

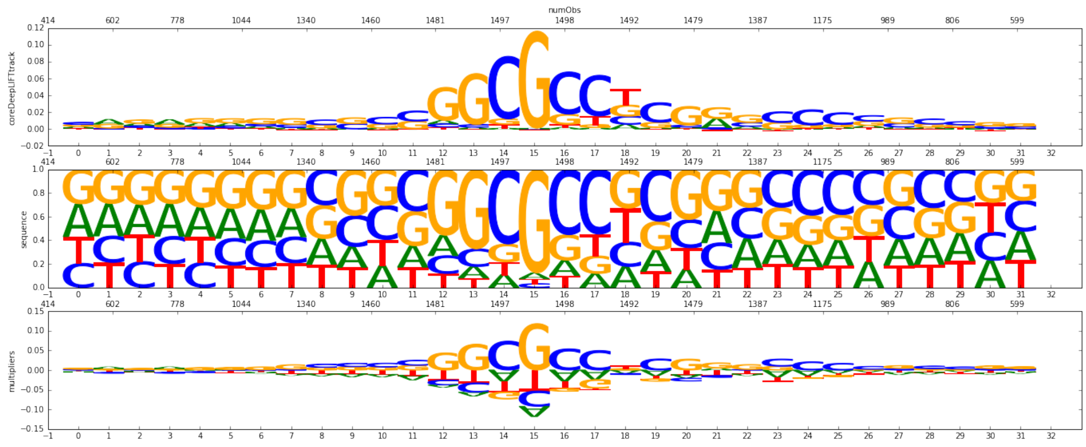


Figure 7: These two motifs were found to be most predictive of true positives in the YA development stage. The descriptions of the three tracks in each motif are given in Figure 5. As in each of the other stages, the GAGA motif was found to be highly correlated with DNA accessibility in the YA stage as well. The second GGCGCCT motif resembles the second motif in the L3 stage, with the difference that the final nucleotide is a T in the YA stage rather than a G in the L3 stage.

might be very close to being significantly upregulated or downregulated, but would be included as ”no change” in our dataset, which might not be the correct classification for that peak. However, to avoid formulating the problem as a regression question, it is necessary to draw the line at some point between peaks that are upregulated/downregulated and have no change. Introducing more classes to capture these middle peaks would reduce the number of peaks in each class, and we would expect that the model would have difficulty learning from such limited number of peaks. However, the limitations of the classification system are something that must be taken into account as a possible limiter of model performance.

We plan to continue extracting relevant motifs for DNA accessibility in *C. Elegans* by first tuning the model to achieve greater predictive performance on the test and validation data. Currently, because the model is overfitting to the training data and not generalizing well to the test data, deeplift only picks up very general motifs that are correlated with the most strong positives (for example, the GAGA motifs). Once the model is better trained, ranking the features again can allow us to find more subtle motifs that differentiate which stage a particular peak would be present in. To tune the model, we might consider experimenting further with hyperparameters (just as we experimented with the stride/width of the maxpool layer in architecture 3). We also might consider changing the window size of the peaks from 200 to something larger, or varying the length of the flanks.

In addition, we plan to run deeplift not just on the stage specific true positives, but also on the positives from the differential tasks (e.g. whether a peak was upregulated/downregulated/no change from EE to L3 to YA). Doing so allows us to discover novel motifs that are correlated with DNA accessibility dynamics in the organism’s development, with the end goal of building the first database of in-vivo regulatory elements in *C. Elegans*.

Bibliography

Brian D. Harfe, Ana Vaz Gomes, Cynthia Kenyon, Jun Liu, Michael Krause, and Andrew Fire. Analysis of a caenorhabditis elegans twist homolog identifies conserved and divergent aspects of mesodermal patterning. *Genes Dev*, 12(16):2623–2635, Aug 1998. ISSN 0890-9369. URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC317087/`. 9716413[pmid].

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015. URL `http://arxiv.org/abs/1502.01852`.

David R Kelley, Jasper Snoek, and John Rinn. Basset: Learning the regulatory code of the accessible genome with deep convolutional neural networks. *bioRxiv*, 2015. doi: 10.1101/028399. URL `http://biorxiv.org/content/early/2015/10/05/028399`.

Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not just a black box: Learning important features through propagating activation differences. *CoRR*, abs/1605.01713, 2016. URL `http://arxiv.org/abs/1605.01713`.