ChromCNN: Chromatin State Annotation as an Image Segmentation Problem

Arnav Ghosh (ag983)

14th December 2018

1 Introduction

Underlying the basic process of many studies, such as those tracking the progression of Melanoma [7], is the need to identify the factors that affect the transcriptional activity of a gene. Consequently, an increasingly important tool to highlight these factors and study human gene regulation in general is chromatin state annotation. However, existing tools that perform these annotations are sometimes inaccurate, inefficient and hard to generalize.

Traditional Machine Learning methods can be categorized into two types: discriminative and generative learning models. As with this task, many problems in computational biology are tackled using the second category. These models, while useful because of our ability to obtain an uncertainty in predictions, are difficult to design and generalize because of the explicit probabilistic assumptions that must be made, which may be complicated or change from task to task.

Thus, for this project, I explore the effectiveness of using a particular class of discriminative models known as Convolutional Neural Networks as an alternative method of chromatin state annotation. The use of CNNs for many computer vision tasks has demonstrated it's propensity to effectively incorporate spatial context, which, given that the state of any genomic bin is not independent of nearby bins, is particularly useful for this task. In this paper, I propose two models that assign chromatin states to the bins present in an input genomic sequence. While my goal is to demonstrate that CNNs can accurately annotate a sequence, I place a particular emphasis on interpreting the features it uses.

2 Background

2.1 Chromatin State Annotation

Sequences of approximately 150 base pairs in the DNA are curled around a center of an eight-protein complex, known as the histone octamer. Each structure and any of its associated proteins is called the Chromatin[1]. These structures enable the attachment of various structural and regulatory proteins and thus, its composition allows us to determine the transcriptional activity of a gene.

The histone proteins contained at the center often undergo various modifications, changing the interaction of this gene with different proteins. Thus, certain combinations of modifications could, among other effects, make regions more likely to initiate the transcription of a protein or encourage the binding of transcription factors that suppress the expression of a gene. Since the 2000s, scientists have discovered over a hundred chromatin modifications (combination of histone modifications) and have

attempted to group them together into a discrete set of states[2], describing whether a region is an **Active Promoter** or a **Weak Transcriber**, for example.

Thus, researchers attempt to annotate each 200 base pair interval in a DNA sequence with one of these states. They do so by looking at the location of the region in the genome and various epigenomic marks associated with it, such as the presence or absence of certain histone modifications. While there continues to be ongoing debate on how many chromatin states should be defined, tagging each region with the role they play during transcriptional activity helps scientists understand the relation between regions of the DNA and protein expression and disease control.

2.2 Image Segmentation & Convolutional Neural Networks

Image segmentation is the problem of partitioning an image into discrete, meaningful segments. The motivation for doing so is that representing an image as human-interpretable regions as opposed to pixels allows for greater reasoning about the content of the image. Most approaches tackle this problem by tagging each pixel as belonging to one of K classes. Pixels belonging to one class make up a single segment in the image (though not necessarily one object).

Over the last decade, Convolutional Neural Networks (CNNs) have found wide use in the Computer Vision community to tackle problems like image classification, object detection and image segmentation. At a high level, each CNN can be thought of having a set number of layers. Each layer takes as input a feature map, performs a single operation (Figure 1) on the pixels of this map and outputs the result of this operation as another feature map. Each feature map can be thought of as some intermediate representation of the initial image, with altered height, width and number of channels. For the purposes of segmentation, the output of this network is a feature map of the same dimensions as the input where instead of an RGB value, each pixel simply holds the class it belongs to.

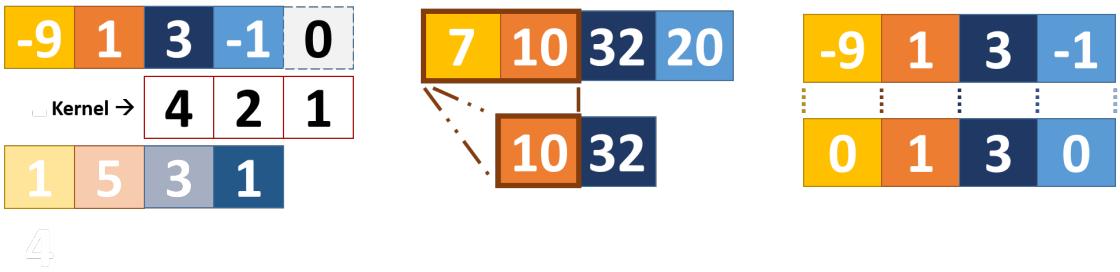

Figure 1: Left to Right: (1) **1D Convolution**: We calculate the 1 in the final cell of output map as follows: (1 * 3) + (2 * -1) + (4 * 0) = 1. The weights of the kernel are learned during the training process. (2) **Max-Pooling**: We slide a 2 x 1 window over the input and place the maximum value of all values in this window in the output map. (3) **ReLU**: Each cell of the input is either preserved if positive or set to 0 if negative.

3 Existing Methods

One of the first and most effective methods for chromatin state annotation is ChromHMM [3]. The model uses a multivariate Hidden Markov Model, wherein each state corresponds to a chromatin state and the emission distribution is a product of independent Bernoulli random variables, each cor-

responding to the presence of a single histone modification. However, the authors of [3] do note a number of drawbacks. In particular, the learning process takes approximately a day and some biological assumptions may be needed for the underlying emission distribution. ChromHMM's accuracy on the validation data we describe in section 4.1 is **0.897**.

4 Proposed Methods

4.1 Dataset

For the experiments below, I use sample data of chromosome 11 drawn from a Erythrocytic Leukaemia (K552) cell line that is available with the ChromHMM package [3]. For each 200 base pair region (bin) in the sequence, the dataset labels the ground truth chromatin state (from a total of 10 possible states) and notes the absence of presence of 9 genetic modifications. Importantly, the use of 10 states differs slightly from the conventional 15 states used to segment a genetic sequence. However, as the authors of [3] point out, the final 5 states appear too infrequently in the sample data and thus, are currently ignored.

- **Modifications**: CTCF, H3K27ac, H3K27me3, H3K36me3, H3K4me1, H3K4me2, H3K4me3, H3K9ac and H4K20me1.
- **States**: Active Promoter (**0**), Weak Promoter (**1**), Poised Promoter (**2**), Strong Enhancer-1 (**3**), Strong Enhancer -2 (**4**), Weak Enhancer (**5**), Poised Enhancer (**6**), Insulator (**7**), Transcriptional Elongation (**8**), Transcriptional Transition (**9**).

Because CNNs can't receive data of arbitrary length, we split the entire genome into 16 bin sections, each corresponding to a single image. The consequences of doing so are briefly explored in section 6.3. To encode the available genetic information into the image, we convert the modifications at each bin into a 9-dimensional vector of 0s and 1s, with each denoting the absence or presence respectively of a particular modification. Thus, we have created images with dimensions 1 x 16 x 9. We represent the intended output at each bin as a 1-dimensional vector of the state assigned to it. Finally, the dataset is split into 33613 training images, 4203 validation images and 4201 test images.

4.2 Network Architectures

4.2.1 Segmentation Net

To create this network, I use an idea proposed in [4], which consists of two sections, an encoder followed by a decoder:

- **Encoder**: This section is responsible for creating a compact representation of the input sequence. It consists of 3 modules. At each module, the incoming feature map from the previous stage undergoes a 1D Convolution, a ReLU and a 1D Max-Pooling operation with a kernel size of 2. Each 1D convolution layer uses twice the number of filters of the stage before it, with the first module using 32 filters. Consequently, we see that after each module, the number of channels in the feature map doubles, whereas the width of the feature map halves.
- **Decoder**: This section is responsible for translating a feature description of the input sequence (generated by the encoder) into pixel-wise annotations. As with the Encoder, this sections also consists of 3 models. At each, the incoming feature map is upsampled by a factor of two and undergoes a 1D convolution.
- **Base-Wise Classification**: At the final layer, we perform 1D convolution with exactly 10 kernels. Convolution with each kernel produces a single channel in the output feature map,

denoting a score for each pixel. We then use the softmax function across channels in this output to generate a probability distribution over classes for each pixel. The class with the highest probability is then assigned to that pixel.

4.2.2 U-Net

This architecture is identical to the one proposed above, with a single modification [5]. The network uses skip connections, where we take a feature map from an early layer and append it to the input feature map of some following layer, completely bypassing any layers in between. In this case, we use three skip connections. The first takes the output feature maps of the first encoder module and appends it to the input feature maps of the final module in the decoder. We do the same with the second encoder module and the second-to-last decoder module and the third encoder module and the third-to-last decoder module.

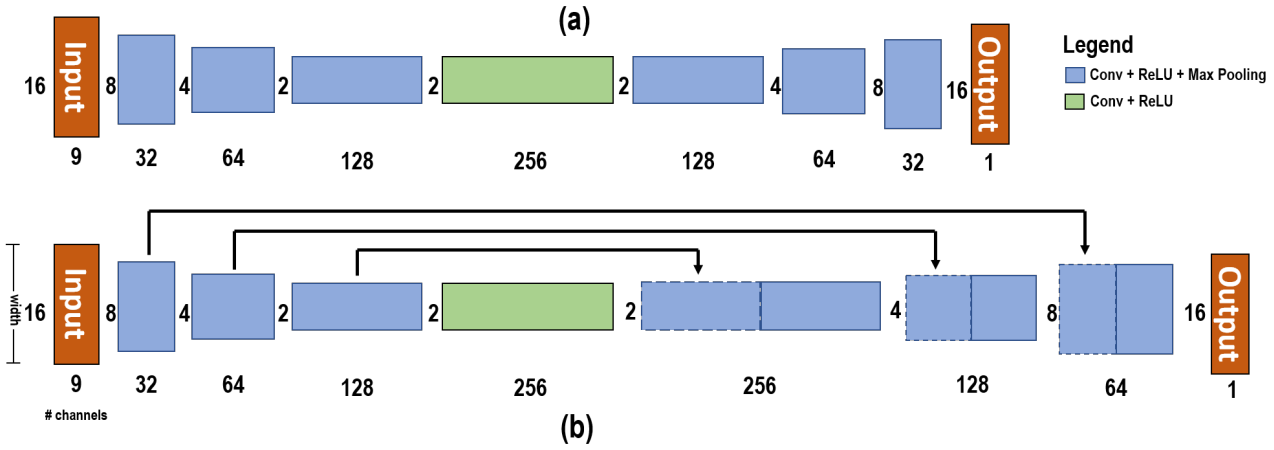

Figure 2: Network diagrams for (a) SegNet and (b) U-Net with arrows denoting the skip connections.

4.3 Training

Both networks were created using Keras, with a Tensorflow backend. A batch size of 32 and the default Adam optimizer were used during training, with the network being evaluated on a held-out validation set after each epoch, for a total of 30 epochs.

The networks aim to minimize the following cross-entropy loss:

$$\frac{1}{n} \cdot \sum_{i=1}^{n} - \sum_{c=1}^{10} y_{i,c} \cdot log(p_{i,c})$$

where n is the number of bins (pixels) in the input, $y_{i,c}$ is 1 if the ground truth class for bin i is c and 0 otherwise and $p_{i,c}$ is the predicted probability that bin i belongs to class c.

5 Results

Figure 3 allows us to make a few observations. To begin with, regardless of the kernel size, U-Net's validation loss is consistently lower than that of the corresponding SegNet model. Thus, the skip-connections clearly are helpful and we explore reasons for this in Section 6.2. We also notice a very

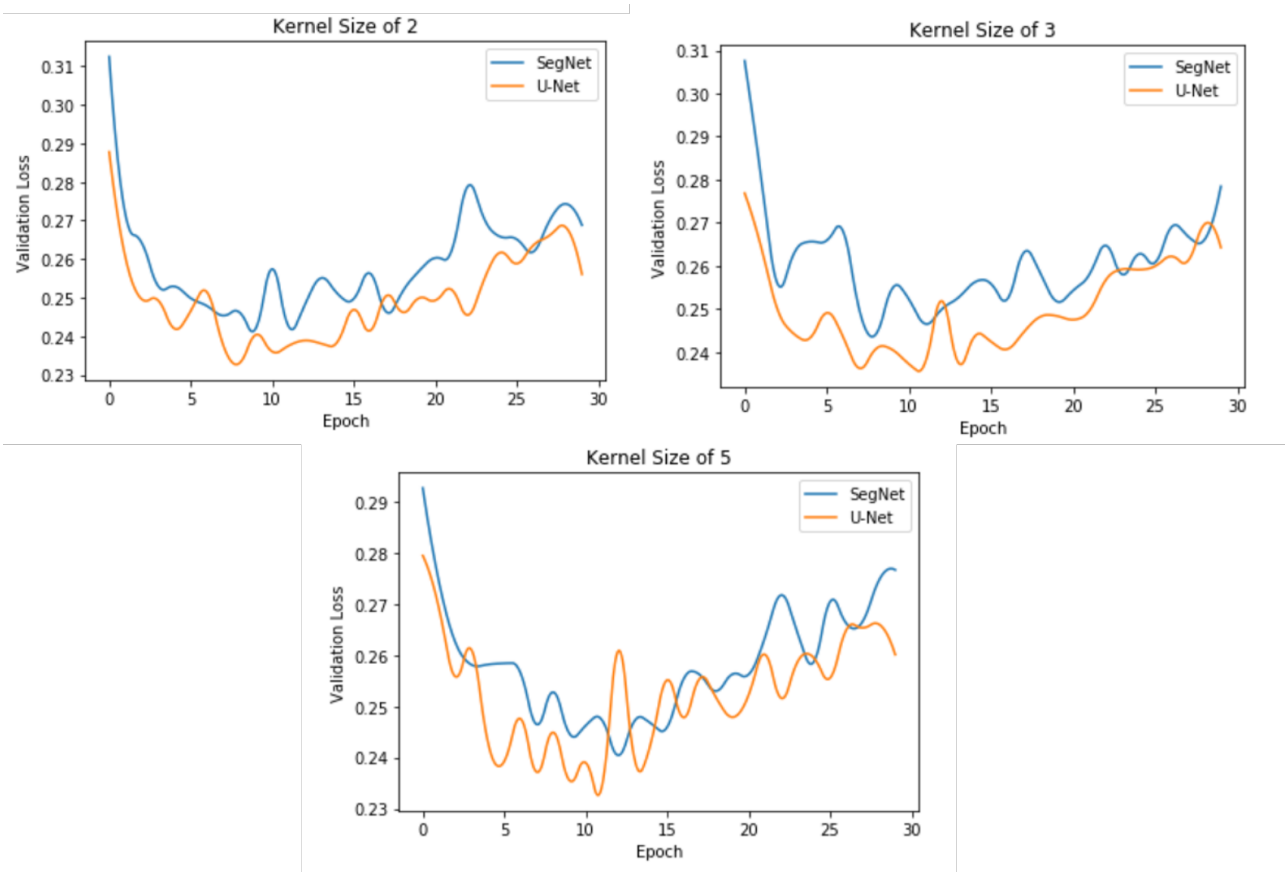

Figure 3: Validation Loss vs. Training Epochs for both architectures using 3 different kernel sizes.

Table 1: The maximum validation accuracy of various models

Model	SegNet			U-Net		
Kernel Size	2	3	5	2	3	5
Max. Accuracy	0.891	0.904	0.899	0.901	0.927	0.915

rapid drop in the loss in the first few epochs. This suggests that there are certain patterns in the data that can be captured fairly easily. However, that the loss doesn't continue dropping further, suggests that there are aspects of the problem that these CNN architectures are not expressive enough to handle and require more complex networks.

Table 1 demonstrates the effect of using different kernel sizes. Interestingly, the effects of different kernel sizes is more pronounced in U-Net, suggesting that in more complex architectures, the choice of kernel sizes is non-trivial. The optimal kernel size is 3 and increasing the kernel size does cause a drop in performance, suggesting that the network is unable to effectively put the additional information provided to use.

Finally, in all the graphs, we notice that the loss tends to reach a minimum and then gradually increases as the model is trained for more epochs. While not plotted, the training loss continued to decrease at this point. This suggests that the architectures begin to learn artifacts that are specific to the training set and don't apply in general, leading to a higher validation loss after a few epochs.

The rise in validation loss occurs more quickly in models that use a initial kernel size of 5 which is reasonable since these models have more parameters than the other two.

6 Discussion

For the experiments below, I used versions of the SegNet and U-Net models of kernel size 3 whose weights provided the maximum validation accuracy over the 30 training epochs.

6.1 Chromatin State Confusion

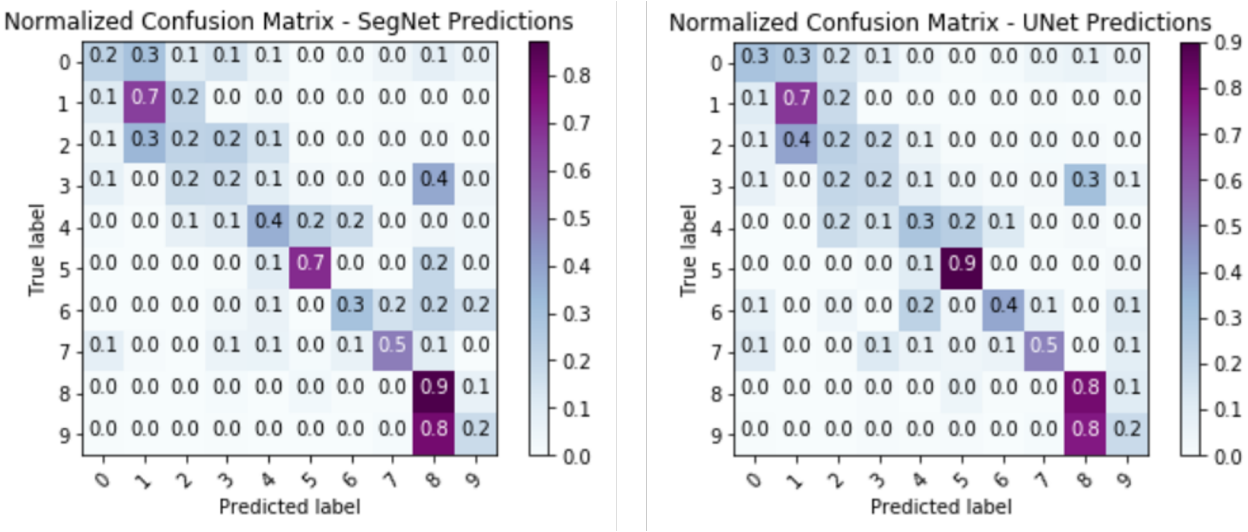

Figure 4: Normalized confusion matrices for predictions made by SegNet and U-Net with an initial kernel size of 3 on the test set.

In Figure 4, for both models, we see that classes 1, 5 and 8 have the highest recall, among all classes. This suggests that the network is capable of distinguishing between the three main classes of chromatin states: promoters, enhancers and transcriptional states, since each corresponds to exactly one of these states. However, at the same time, we note that the recall within these main categories can be improved. Take classes 0, 1 and 2, which correspond to Active, Weak and Poised Promoters. For both Active and Poised promoters, both models are marginally more likely to label bins as Weak Promoters than their true label. Thus, since the pattern of histone modifications for bins with these labels is similar, the model finds it slightly difficult to distinguish between them and favours annotating any bin displaying these patterns as a Weak Promoter.

Finally, the most intriguing result from Figure 4 is the markedly poor performance for class 9, Transcriptional Transition, where most bins are annotated as belong to class 8, Transcriptional Elongation. The above point only partially explains this effect. The more compelling reason is there is an imbalance between the number of examples available for class 8 and 9, with 10 times more bins corresponding to the former. Thus, given the similarity in the histone modification patterns for these examples, the best way for the network to minimize our loss function would be to focus on correctly differentiating all the other classes and simply annotating all examples belonging to classes 8 and 9 as the former.

6.2 Impact of Using Skip Connections

As seen from the validation accuracies and confusion matrices, skip connections (in U-Net) improve our segmentations. One potential reason for this improvement is that these connections provide more 'fine-grained' predictions. Take the Strong Enhancer-2 state (class 4). This state seems to occur in very short stints across the genome [3], making it more susceptible to being mislabeled by coarse segmentations. As Figure 4 indicates, U-Net performs slightly better at annotating this class than SegNet, validating the idea that it's segmentation is more fine-grained. This property can also be visualized in Figure 5, where U-Net retains the sole class 4 and 7 labels whereas SegNet does not and instead expands the class 6 segmentation.

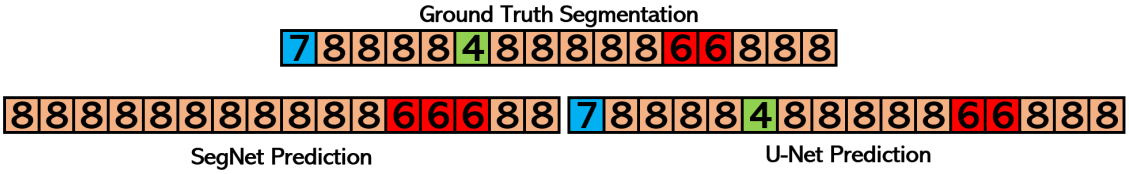

Figure 5: Sample predictions by the two models indicating the coarseness of their segmentations

6.3 Occlusions

An effective way to interpret the cues used by the CNN to perform any classification is to occlude part of the input and observe the results [6]. For this problem, I sample a DNA sequence from the held-out test set that contains a bin with ground truth class 8 at its center and atleast one histone mark at each bin (called region in Figure 6). I then iteratively occlude a single bin, by indicating the absence of all histone marks. At each iteration, I observe the change in the predicted probability of the same center bin belonging to class 8 against the same sample with no occlusion.

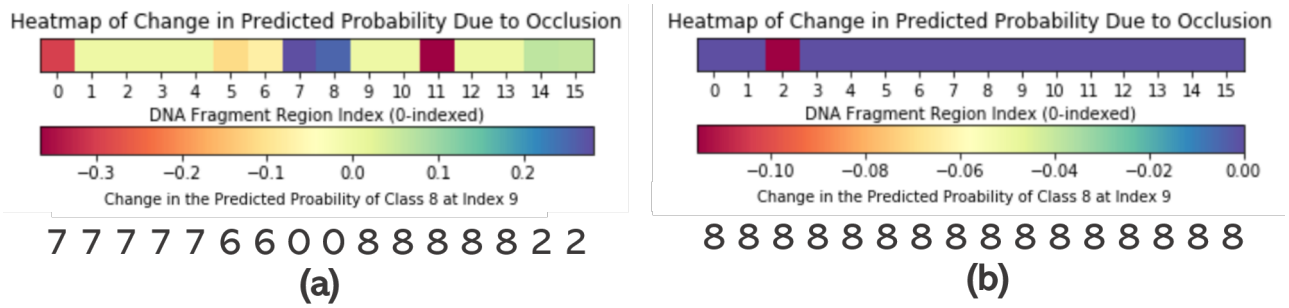

Figure 6: The results of occluding DNA fragment regions (bins) in two different data samples. The heatmap value at each index i indicates the change in the predicted probability of class 8 at index 9 if bin i was occluded. Ground truth classes are provided for reference in the final row. When passed through the network, the un-occluded samples yield the ground truth result. **Note:** The heatmap value at index 9 provides no information because it's probabilities are the ones being observed.

That we achieve high precision and recall in segmenting regions of this class allows us to be confident of the following conclusions. In (a), we see that as we consider regions further behind the

region of interest, the effect on the latter's probability distribution slowly tapers off but is nonetheless significant. Thus, we see that one of the reasons using a CNN is effective is because it takes into account a large context window. In (b), we see that altering the regions immediately preceding bin 9 has virtually no impact on the probability predicted by the CNN. In conjunction with (a), this suggests that the CNN doesn't just rely on a larger window of bins but also uses global features in making predictions.

Finally, these observations also underline one of the flaws of using a CNN. In (a), we see that occluding the first region affects the probability significantly. Because the four regions after that have no effect on the probability, we can conclude that this is because this was where the DNA segment was split to pass into the CNN as opposed to any real semantic relation to region 9. Thus, this suggests that splitting the DNA sequence does affect the probabilities predicted at regions that seem to be 'far' away from the split. Any extension would naturally have to remedy this.

6.4 Transfer Learning

In practice, researchers are unlikely to have access to as much labeled data for one particular genome as we did for this project. Thus, a crucial criterion for using these architectures would be their ability to adapt to annotating new cell lines, with minimal additional data. Thus, for this experiment we take our U-Net architecture, pretrained on the K562 cell line and allow for additional training on 100 training examples for the annotation of chomrosome 11 of the B-Lymphocyte (GM12878) cell line. We then test this new network's ability to annotate unseen examples from the GM12878 cell line.

Training Process	Without Additional Data	With 100 Additional Examples
Test Accuracy	0.876	0.935

Table 2: Test accuracy on the GM12878 cell line with and without finetuning

As seen from Table 2, the model can easily be applied to other datasets. The slight drop in test accuracy if the model is not fine-tuned is indicative of the fact that there are patterns that apply across cell lines and those that are specific to each. Furthermore, that we get a 0.6 increase in accuracy with little additional data demonstrates the model's flexibility and readiness for real use.

7 Conclusion

In summary, we show that using Convolutional Neural Networks provides a marginal improvement over ChromHMM in annotating chromatin states. In particular, these networks make effective use of both local and global cues in DNA segments to perform such annotations. We find that the use of skip connections to provides more fine-grained annotations and that such pre-trained networks can easily be applied to other datasets. Thus, we have created a tool that allows researchers to effectively, with limited data, annotate chromatin states.

However, despite these benefits, there are certain improvements that could be explored. In particular, ways to remedy the length restriction we impose on the DNA segments and methods to improve annotation accuracy of similar categories would be interesting experiments to try.

8 Source Code

Code used to prepare training data, define, train and analyze models is available here.

References

[1] Guillaume J. Filion, Joke G. van Bemmel, Ulrich Braunchschweig et. al. *Systematic Protein Location Mapping Reveals Five Principal Chromatin Types in Drosophila Cells.*

[2] Monya Baker. *Making sense of chromatin states.* https://www.nature.com/articles/nmeth.1673

[3] Jason Ernst and Manolis Kellis. *ChromHMM: automating chromatin state discovery and characterization.* https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3577932/

[4] Jonathan Long, Evan Shelhamer and Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation.* https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

[5] Olaf Ronneberger, Philipp Fischer, Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation.* https://arxiv.org/pdf/1505.04597.pdf

[6] Martin Thoma. *Analysis and Optimization of Convolutional Neural Network Architectures.* https://arxiv.org/pdf/1707.09725.pdf

[7] Petko Fiziev. et. al. *Systematic Epigenomic Analysis Reveals Chromatin States Associated with Melanoma Progression.* https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5473172/