CS230

Denoising ATAC-seq with Convolutional Neural Networks

Nic Fishman and Sarah Gurev
Department of Computer Science
Stanford University
njwfish@stanford.edu, sgurev@stanford.edu

Abstract

Chromatin is a compressed version of DNA that can be opened or closed, which is important because a gene cannot be expressed when chromatin is in its closed form. ATAC-seq is a sequencing technique used to determine what parts of chromatin are opened or closed. However, the quality of ATAC-seq data is heavily dependent on many environmental factors, and therefore prone to noisiness. Our goal was therefore to create a model to denoise low quality ATAC-seq data. We used a NASH architecture search to develop a CNN architecture that was trained on high quality, low quality pairs of data. Our final model is able to reliably convert noisy data into high quality data.

1 Introduction

ATAC-seq is a sequencing technique that is able to determine what parts of chromatin are in its open or closed form (Figure 1). This information is useful to determine gene expression, since whether a gene is able to be expressed depends on the chromatin accessibility. This technique is rapidly gaining popularity due to its speed and low requirements on the biological sample as well as its high signal to noise ratio. However, the quality of ATAC-seq data is impacted by several environmental factors, including the amount of input DNA and sequencing depth. ATAC-seq data is therefore prone to noisiness. As a result, we have used an architecture search to develop a CNN architecture to denoise ATAC-seq data. The input to our algorithm is low quality data. We then use a CNN to output a predicted denoised conversion of the data. Our algorithm learns for a specific cell line based on the probability of a peak at one location, what is the probability chromatin will be open at another location. Because of the biological relevance of the data, within a cell line, there is a lot of information that can be learned about the relative locations of peaks.

2 Related work

Developed in 2015, ATAC-seq is a relatively new biological tool. As such, there has been no published effort to denoise ATAC-seq data. Currently, instead of denoising ATAC-seq data, bioconductor packages like ATACseqQC have been developed to assess read quality, preprocess and assess datasets, and identify high-quality datasets to avoid drawing conclusions from low-quality ATAC-seq experiments.

Our methodology instead is based on Koh et al.'s work to denoise ChIP-seq data, a similar chromatin-based sequencing technique. Koh et al. developed Coda, a convolutional denoising algorithm, to denoise three different sources of noise. It trains two separate CNNs, a regression task to predict signal and a binary classification task to predict the presence or absence of a peak. This approach

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

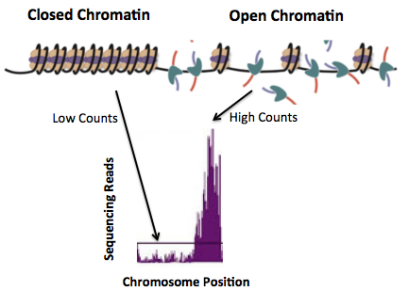


Figure 1: ATAC-seq signals depict chromatin accessibility.

was able to substantially improve the quality of new ChIP-seq data, though Coda's performance depends strongly on the similarity of the noise and data distributions between the training and test sets. Additionally, Coda's performance depends on noise parameters in the test data already being known, so work still needs to be done to make Coda more robust and generalizable.

Furthermore, there has been work done to denoise sequencing data, through algorithms like DUDE-seq. Ostensibly, denoising sequencing data would help to denoise the ATAC-seq data which relies on sequencing results. DUDE (Discrete Universal DEnoiser) works to reconstruct sequences with finite-valued components corrupted by a noise mechanism that corrupts each symbol independently and statistically identically, which can be applied well to the noise model of DNA sequencing. DUDE is a two-pass algorithm that first collects a statistics vector of the count of the occurrence of each base and afterwards finds a reconstruction symbol (base) that minimizes the expected loss with respect to the empirical estimate.

In terms of methodology, our work is based on Elsken et al.'s method to automatically search for well-performing CNN architectures using a hill climbing procedure. Neural Architecture Search by Hillclimbing (NASH) works by applying network morphisms followed by short optimization runs by cosine annealing. The network morphisms are based on the work of Chen et al.'s Net2WiderNet and Net2DeeperNet, in that a change is made to a base network such that the changed neighbor network has the same loss as the base network. The strength of this method is it yields competitive results, even though it only requires computational resources on the order of magnitude of training a single network. Additionally, another clever technique, as introduced by Loshchilov et al., is cosine annealing with warm restarts. Warm restarts help improve the rate of convergence for training deep neural networks.

3 Data Processing

We obtained high quality ATAC-seq reads from the publicly-available GEO sample SRR891270. We aligned these reads to the HE38 Chromosomes. We then converted the read counts to a BAM file. To obtain corresponding noisy datasets, we subsampled the BAM file to 10% of the original using samtools. This is the same methodology used to analyze noise in the ATACseqQC paper. We then called peaks on both the high quality and noisy BAM files by converting the files to narrowPeak format using MACS2 with a low threshold (Figure 2). The peak calls were binned into equal size bins of 25 base pairs and given the value of the maximum signal in each bin.

For each example, we uniformly sampled from the binned dataset, taking a consecutive stretch of 4096 binned peak values, where the center 1024 values are the ones to be predicted. We used a Hilbert Curve to map the one dimensional list of the sampled binned values to two dimensions, while maintaining positional relevance.

In total, we had 9753 Training (5% of which is Validation) and 972 Test examples. Below you can see a graphed representation of the binned peak values (Figure 3).

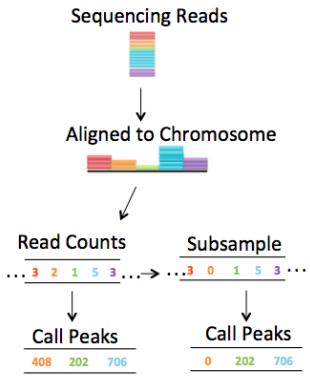


Figure 2: An Overview of Data Processing Steps.

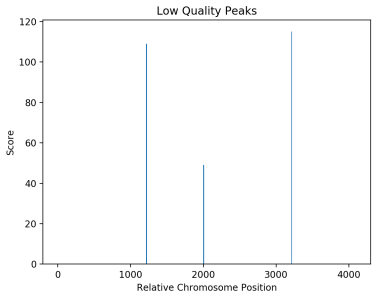


Figure 3: An example graphed input of Low Quality Data. The center 1024 values are the values to be predicted by our model.

Model	Avg MSE Loss
1Dconv random search	250.7562382
2Dconv random search	278.4532116
2Dconv + Hilbert random search	150.1562584

Figure 4: Average MSE Loss for 1D, 2D, and 2D mapped to a Hilbert Curve inputs.

4 Methods

We began with a random architecture search for the best initial model, starting with one Conv2D layer going into one Dense layer. We randomized Kernel size, filters and hidden units. We began by running this random search ten times for twenty epochs each and calculating the average MSE Loss for a 1D input, but this had poor performance. We then moved to doing the same search, but for data arranged as 2D input, but the performance was worse. We finally decided to perform this search on data mapped to a Hilbert curve, and the performance significantly increased (Figure 4). A Hilbert Curve manages to map a 1D input to 2D while maintaining positional relevance of the data. After selecting the initial model (seen in Figure 6) which uses the 2D Hilbert Curve Mapping as an input, we conducted Neural Architecture Search by Hillclimbing, which uses a Greedy Hill Search to find a new best model after randomly changing the parent. It does so by applying network morphisms to develop 8 new models.

Let $N(X)$ denote a set of neural networks where $X \subset \mathbb{R}^n$. A network morphism is a mapping $M : N(X) \times \mathbb{R}^k \to N(X) \times \mathbb{R}^j$ from a neural network $f^w \in N(X)$ with parameters $w \in \mathbb{R}^k$ to another neural network $g^w \in N(X)$ with parameters $w \in \mathbb{R}^j$ such that $f^w(x) = g^w(x)$ for every $x \in X$. In this way, a network morphism is a change to the current model such that the change does not impact the current loss of the model. These changes can be making a layer of the model deeper or wider, which are based off Net2WiderNet and Net2DeeperNet. We can see Net2DeeperNet works by the new layer implementing the identity function, so that the output of the new model stays the same. By not changing the loss of the model, we can continue training with the updated model without starting over in terms of loss (training only 10 more iterations will give a good representation of the performance of the model). We then train each child network with cosine annealing for ten epochs. Cosine annealing works by decaying the learning rate for the i-th run as follows:

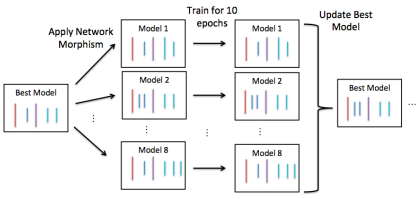


Figure 5: The Steps of NASH

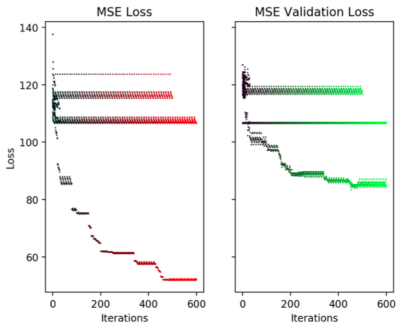


Figure 7: MSE Loss is graphed at each epoch for Training and Validation Sets.

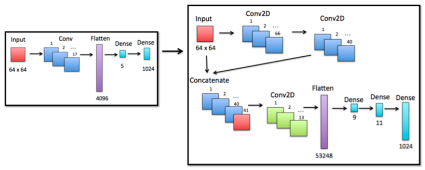


Figure 6: The initial model is changed to the final model after 60 iterations of NASH.

Iterations of Search	Train Error (9753 samples)	Test Error (972 samples)
0	150.156	128.45
20	137.878	122.953
40	92.135	90.574
60	89.778	88.805

Figure 8: MSE Loss for Training and Test Sets at iterations of Architecture Search.

$$\eta_t = \frac{1}{2}(\eta^i_{start} - \eta^i_{end})(1 + cos(\frac{T_{cur}}{T_i}\pi)), \tag{1}$$

where η^i_{start} and η^i_{end} are the ranges for the learning rates and T_{cur} accounts for how many epochs have been performed since the last restart and T_i is the number of epochs until a new warm-started run or restart of SGD is performed.

The most promising child (the child with the best validation loss) becomes the new best model, and the process is repeated (Figure 5). It is important that the child selected to be the new best model can be the original model itself, which keeps performance from decreasing.

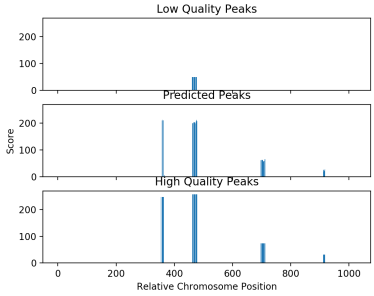
Our final model was selected after 60 iterations (seen in Figure 6). It involves two Conv2D layers, the second of which is concatenated with the input and fed into another Conv2D layer. The result is then flattened and fed through three Dense layers. The final layer outputs 1024 values, which are the predicted values denoised values.

5 Experimental Results

For our model's hyperparameters, we chose the default values used in Elsken et al.'s paper: a η_{start} of 0.05 which is annealed to a η_{end} of 0.0 after $epoch_{neigh}$ of 17 epochs for cosine annealing, 100 training epochs, and 8 children of each base network for the Architecture Search. We used MSE Loss to evaluate our models. Shown in Figure 7 is the MSE Loss for Training and Validation Sets over each epoch of training during the Architecture Search. You can see where several times the model reaches a threshold with no improvement (cases where no child of the base network improves performance after training for ten epochs), and the training restarts again.

Shown in Figure 8 is the Training and Test Loss for models at several iterations of the Architecture Search. We can see the MSE Loss improve as the search progresses for both Training and Testing error as well as notice as Training and Testing error get closer to one another.

Qualitatively, we can see that our model usually does a good job with predicting the correct location on the chromosome that should be open as well as the correct score for those locations (the confidence that the chromatin is open there). We can see this in the examples in Figure 9.



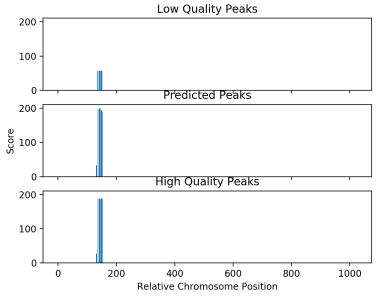


Figure 9: Predicts correct location and score. Shown here are two examples of Low Quality Data, Predicted High Quality Data, and True High Quality Data.

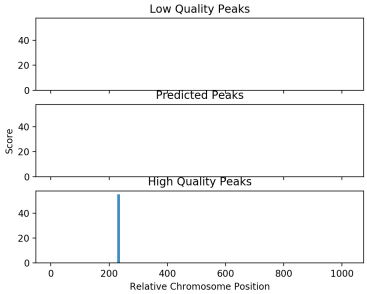


Figure 10: Predicts all chromatin to be closed in case where Low Quality has no nonzero values.

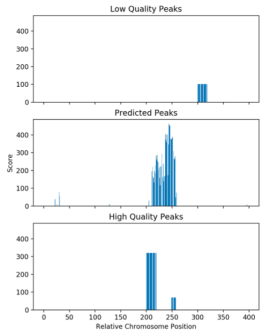


Figure 11: An incorrect example, but Predicted is still shifted right towards true location and score is increased to be more similar to true confidence.

We can also see when there is no information given in the low quality data, it predicts that there is nothing, even when the High Quality data may have some values (Figure 10). This is an expected outcome. Furthermore, sometimes the exact values of the score and position are not correct, but we can see that the score is generally more similar to the true High Quality Data, and that the position shifts towards the true High Quality Data (Figure 11). This is an example of overfitting the training data, since it identifies a favorable pattern for the data, but is ultimately not entirely correct. We otherwise managed to mostly mitigate overfitting by selecting our new best model during the architecture search using validation rather than training loss. By limiting the number of iterations of the architecture search, we prevent there from being enough iterations run to overfit the validation set.

6 Conclusion

We performed a NASH architecture search to develop a CNN architecture that is trained on high quality, low quality pairs of data. Our final model was trained on 60 iterations of the architecture search, and converts noisy data into high quality data with an Test MSE Loss of 88.805. Our algorithm learns for a given cell line the probability chromatin will be open at a location given the other known locations of open chromatin nearby. Because of the biological relevance of the data, within a cell line, there is a lot of information that can be learned about the relative locations of peaks, which is what makes solving this problem possible. Moreover, our computationally simulated noise, done via subsampling, is a reasonable estimate of ATAC-seq noise caused by environmental factors such as amount of input DNA and sequencing depth, so our model should do a good job of denoising true low-quality data.

In the future, we would develop a pipe-line for cell-line specific training. We would also attempt to confirm the biological relevance of our denoised data using properties of chromatin accessibility for evaluation. We would also replicate our work with other sources of noise.

7 Contributions

Nic was responsible for implementing the architecture search and data processing. Sarah was responsible for data processing, related work, and writing the paper.

References

[1] Chen, Tianqi, Ian Goodfellow, & Jonathon Shlens. "Net2net: Accelerating learning via knowledge transfer." *arXiv preprint arXiv*:1511.05641 (2015).

[2] Chollet, François. "Keras (2015)." (2017).

[3] Elsken, T., Metzen, J. H., & Hutter, F. (2017) Simple and efficient architecture search for convolutional neural networks. *arXiv preprint arXiv*:1711.04528.

[4] Koh, P. W., Pierson, E., & Kundaje, A. (2017) . Denoising genome-wide histone ChIP-seq with convolutional neural networks. *Bioinformatics*, **33**(14):I225-I233.

[5] Lee, Byunghan, et al. "DUDE-Seq: Fast, flexible, and robust denoising for targeted amplicon sequencing." *PloS one* 12.7 (2017): e0181463.

[6] Loshchilov, I., & Hutter, F. (2016) . Sgdr: Stochastic gradient descent with warm restarts. *Proceedings of the International Conference on Learning Representations* (ICLR).

[7] Ou, Jianhong, et al. "ATACseqQC: a Bioconductor package for post-alignment quality assessment of ATAC-seq data." *BMC genomics* 19.1 (2018): 169.

[8] Sos, Brandon Chin, et al. "Characterization of chromatin accessibility with a transposome hypersensitive sites sequencing (THS-seq) assay." *Genome biology* 17.1 (2016): 20.