DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences

Team DanQ: Megan Carlson, Phila Dlamini, Jesalina Phan, Rosy Sun

Introduction

Despite much emphasis in literature on the protein-coding regions of the human genome, over 98% of it is non-coding. Much of this non-coding DNA plays key roles in gene regulation, and around 93% of disease-associated genetic variants lie within these regions. This makes it critical to understand the functional roles of non-coding sequences to gain insight into the genetic and epigenetic basis of human health.

DanQ is a deep learning model that attempts to predict the function of non-coding DNA directly from its sequence. It uses a hybrid architecture that combines convolutional neural networks, which detect local motifs, with bi-directional recurrent layers, which capture longer-range dependencies between upstream and downstream motifs. This design enables the model to learn both the building blocks and the regulatory grammar of gene expression. The problem is formulated as a multi-label classification task: given a DNA sequence, the model outputs a binary vector indicating the presence or absence of 919 different epigenetic marks. We were drawn to this project because it sits at a fascinating crossroads between biology and machine learning. The idea that a neural network can extract meaningful biological insight purely from DNA sequences is incredibly exciting. Rebuilding DanQ gives us the chance to explore how deep models can learn regulatory logic from scratch—and challenges us to think critically about the limitations of data-driven biology.

Methodology

The original DanQ paper utilized human transcription factor binding data from 919 ChIP-seq datasets. With so much large genomic data, it took many days and lots of computational power to train. Instead, we opted to use data from mice. The mouse, *Mus musculus*, is a common model organism in the field of genetics, so there are lots of publicly accessible datasets available to analyze. We pulled the mm10 assembly of the mouse genome from the UCSC genome browser (Perez et al., 2025). We got ChIP-seq transcription factor binding data from the ENCODE project with the following identifiers: ENCFF223ASW, ENCFF228IWF, ENCFF841DLH, ENCFF941FDR, ENCFF893PQE, ENCFF031DDU, ENCFF262ITC, ENCFF281FXQ, ENCFF399UKJ, ENCFF734ZCR, ENCFF625UEM, ENCFF950CJS, ENCFF527VCY, ENCFF791EKG, ENCFF196FFI, ENCFF968IBZ, ENCFF235YHK, ENCFF611SZM, ENCFF981ENW, ENCFF254IPE, ENCFF453OVN, ENCFF510QZL, ENCFF627TGO, ENCFF589AAH, ENCFF672FKP, ENCFF980MPW, ENCFF044OUI, ENCFF067LMI, ENCFF573INC, ENCFF416TBW, ENCFF443MGH, ENCFF567VSU, ENCFF468PLO, ENCFF525MNL, ENCFF700PEG, ENCFF813PFO, ENCFF066CEV, ENCFF839UAT, and ENCFF911NHJ (Luo et al., 2020). These were all samples taken from the Mouse erythroleukemia (MEL) cell line, which was the cell line with the most data available on the portal.

The data was preprocessed by splitting the fasta file of the entire genome into 200 bp bins. We then overlapped this with peaks called in the ChIP-seq data. Any bins that did not overlap at least one peak were discarded. We encoded the presence or absence of a peak for a particular transcription factor in a 39x1 vector, where each position in the label vector corresponded to a specific transcription factor. After identifying bins and their labels, each bin was extended an additional 400 bp upstream and 400 bp downstream to generate a total of 1000 bp in each sequence. This extension enables the model to capture longer-range interactions between cis-regulatory elements.

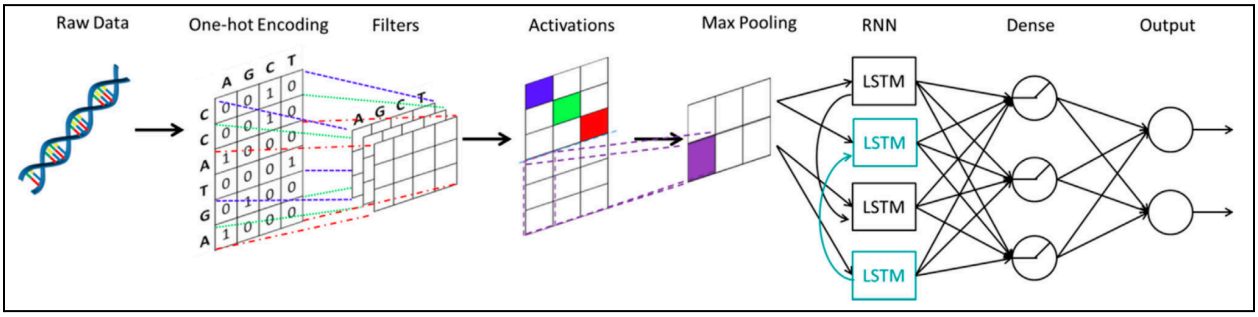
The primary architecture of DanQ is a hybrid model combining convolutional neural networks (CNNs) and bidirectional long short-term memory (BLSTM) layers.This architecture is designed to capture both local patterns in DNA sequences (motifs) and long-range dependencies between them, reflecting the complex structure of gene regulation.

Functionally, the CNN layer in DanQ is responsible for scanning the input DNA sequence to identify short, recurring patterns known as motifs. These motifs are typically indicative of regulatory elements in the DNA, such as transcription factor binding sites. Using convolutional filters allows the model to learn these motifs in a position-invariant way, meaning it can recognize them regardless of where they appear in the sequence

The BiLSTM layer follows the CNN to capture long-range dependencies between the motifs identified by the convolutional layer. Unlike CNNs, which focus on local patterns, BiLSTMs process the sequence in both forward and reverse directions. This bidirectional processing allows the model to understand how motifs interact with each other, considering both the order and spacing of motifs across the sequence. This helps the model learn more complex regulatory patterns that depend on the arrangement of motifs, which is essential for predicting the function of non-coding regions of the genome.

These outputs are then flattened and passed through a fully connected dense layer. A final pass through a sigmoid layer eventually outputs a 39-dimensional output vector, where each element represents the probability of a specific transcription factor acting on the input sequence within that bin.

Our re-implementation applies 320 convolutional filters in the convolution layer, with weights initialized uniformly between -0.05 and 0.05 and biases set to zero. The model is trained using the Adam optimizer with a batch size of 200 and a multi-task binary cross-entropy loss. Dropout is applied after the max pooling and BLSTM layers for regularization.



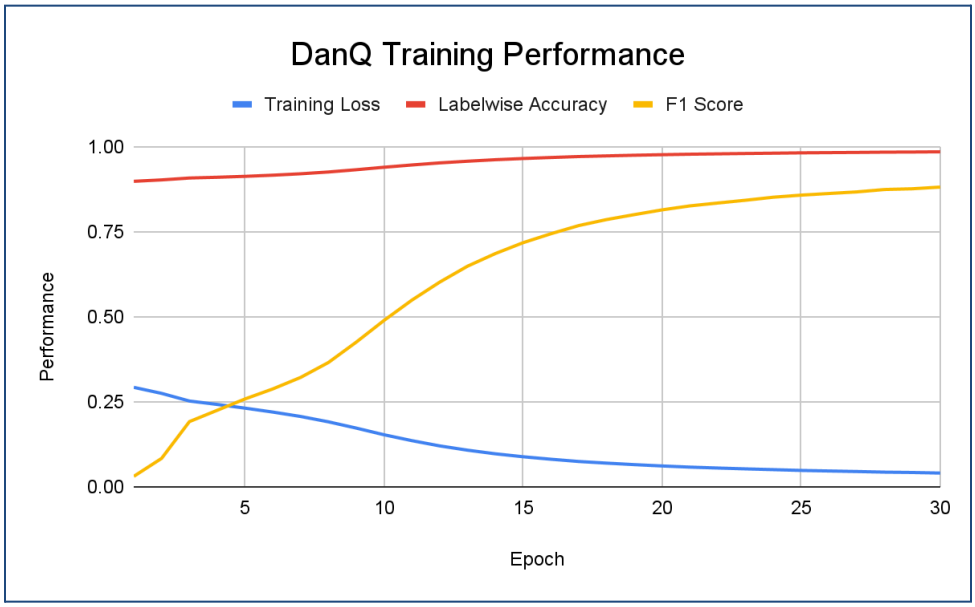
DanQ Architecture (Shen et al., 2022)

Results

Whereas the original paper trained for 60 epochs, we instead decided to train for 30 epochs given our lower quantity of data. We noticed that the label-wise accuracy started very high, around 90%, which can be attributed to the heavy class imbalance in our data; the label vectors frequently consisted of only one or two 1s and remainder were 0s, so a model that predicted all zeros was able to perform well.

To combat this, we instead relied on F1 scores, which combine precision, or how many positive predictions the model made correctly, with recall, or how well the model did at finding all of the positive predictions in the label. This produced a much better metric for us to analyze, as it more rigidly enforced the behavior we desired.

Our training process was largely successful. As shown in the figure below, over the course of our training epochs our label-wise accuracy and F1 scores increased steadily, and our loss decreased. We used binary cross entropy loss for this application, as it works well for our use case of multi-class classification tasks.



DanQ training performance over time

After 30 epochs, we reached a F1 score of around 0.88, which we decided was large enough to indicate that our model was sufficiently well-trained.

Average loss	0.041
Label-wise accuracy	0.985
F1 Score	0.881

DanQ training performance after 30 epochs

Finally, we tested our model. Unfortunately, it seemed to have overfit the data, resulting in a much lower final performance than we had hoped for.

Average loss	0.397
Label-wise accuracy	0.905
F1 Score	0.301

DanQ testing performance after 30 epochs

Challenges

The first major challenge we encountered was preprocessing the data. We were working with many large files, so we had to get creative with the way we were sharing and working with them. Our original preprocessing script proved highly inefficient, and took around 3 days of continuous running to process only 25% of the data. This culminated in a VS Code crash that corrupted Phila's file system and necessitated a total OS reinstall. We then moved things over to Oscar, where the ability to run tasks in parallel greatly sped up our processing and allowed us to finish generating our model inputs.

During development, we encountered an issue where pandas occasionally misinterpreted the 39-character binary target strings (e.g., `"010000...00"`) as floating-point numbers, likely due to the string being parsed as scientific notation. This caused our DataLoader to fail, as it expected an iterable string of binary characters. Despite attempts to resolve the parsing behavior directly, we were unable to consistently prevent the misreading. We even wrote a validation script that confirmed all entries in the CSV files were correctly formatted prior to loading, yet the issue persisted. As a temporary fix, we implemented a validation check in the __getitem__ method of our SequenceDataset class to detect malformed entries and substitute them with a valid dummy sequence and target. Given more time, we would have preferred to identify and resolve the root cause within the pandas loading process itself to avoid relying on post-hoc corrections.

Finally, another major issue with our project is a lackluster final accuracy metric. Our suspicion is that the complex architecture of the original model severely overfit our simplified data; where the original paper used an architecture and hyperparameters to account for 919 transcription factors, we opted to use 39 transcription factors, which was the entirety of those available for the cell line we selected. We kept all model parameters the same as the original model in an attempt to faithfully recreate the results, but in retrospect we should have considered tuning hyperparameters to generate a more appropriately-sized model. By the time we ran our final model in its entirety, we didn't have much time left for hyperparameter tuning. Given more time, we would love to experiment with things like the number of filters in the CNN or the dropout rate in hopes of increasing testing accuracy.

Reflection

Despite challenges along the way, our project ultimately turned out well! We certainly met our base goal of implementing a functional model. Target and stretch goals are a little harder to determine as our final accuracy wasn't great, but we believe that with a few small tweaks we'd easily be able to reach the accuracy listed in our stretch goal.

Despite the overfitting issue, our model did seem to work in the way we intended. As shown in the very high F1 scores seen during training, the model was able to produce quite good predictions based on the input sequences alone. It was exciting to see that this model was able to adapt to data from a totally different organism than the original implementation. This indicates a certain degree of high-level regulatory conservation across mice and humans; the fact that this architecture designed to capture human motif-based regulation worked so well on mice implies similar mechanisms between the species. While it could be improved upon, it was exciting to see our model working in the way we intended for it to!

Our approach changed in a few small ways as we worked on the project, but we ultimately stuck pretty closely to our original plan. For one thing, our Oscar usage ended up playing out differently from what we expected. We, perhaps naively, thought that preprocessing would not take up too much computational load, so we initially attempted to run it locally. This proved too inefficient, so we moved that task over to the HPC. On the other hand, we thought that we'd certainly need to operate on Oscar for our training, but that proved easier to work with on Colab! Training was actually quicker than we expected based on the estimates from the original paper, which was a pleasant surprise. We also slightly pivoted our interpretation of model accuracy when we noticed the effect the class imbalance was having on our results. The shift to focusing on an F1 score proved beneficial as it provided us with a better understanding of how our model was actually doing.

If we had more time to work on this project, we would certainly spend time tuning the hyperparameters to make a slightly smaller model. We shied away from changing almost anything about the initial model's architecture in pursuit of staying true to the original implementation, but tweaking things like convolution channel numbers or hidden layer sizes would likely improve our testing performance. We also should be testing our model during each epoch rather than at the end, which would allow us to monitor overfitting based on loss trends and adapt more quickly if changes are needed. Another cool area to take this project in would be to do some model interpretation to see which motifs are being used to make certain predictions. The ability to pull out common motif patterns from the data opens up a realm of biological applications which could then be explored in a wetlab context.

We found this project to be challenging but ultimately very rewarding. It was a nice challenge to have to source and preprocess our own data, which is something we haven't had to deal with as much throughout the rest of the homework assignments in this course. While difficult, it gave us a more realistic feel for what it takes to create a project from scratch. We also had to deal with new challenges in anticipating the computational load of our jobs and modifying the way we ran things accordingly. It was cool to apply the deep learning tools we've learned

about this semester to a practical application that has far-reaching biological implications, which made the project ultimately highly rewarding.

Works Cited

Luo, Y., Hitz, B. C., Gabdank, I., Hilton, J. A., Kagda, M. S., Lam, B., Myers, Z., Sud, P., Jou, J., Lin, K., Baymuradov, U. K., Graham, K., Litton, C., Miyasato, S. R., Strattan, J. S., Jolanki, O., Lee, J.-W., Tanaka, F. Y., Adenekan, P., … Cherry, J. M. (2020). New developments on the Encyclopedia of DNA Elements (ENCODE) data portal. *Nucleic Acids Research*, *48*(D1), D882–D889. https://doi.org/10.1093/nar/gkz1062

Perez, G., Barber, G. P., Benet-Pages, A., Casper, J., Clawson, H., Diekhans, M., Fischer, C., Gonzalez, J. N., Hinrichs, A. S., Lee, C. M., Nassar, L. R., Raney, B. J., Speir, M. L., van Baren, M. J., Vaske, C. J., Haussler, D., Kent, W. J., & Haeussler, M. (2025). The UCSC Genome Browser database: 2025 update. *Nucleic Acids Research*, *53*(D1), D1243–D1249. https://doi.org/10.1093/nar/gkae974

Quang, D., & Xie, X. (2016). DanQ: A hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences. *Nucleic Acids Research*, *44*(11), e107–e107. https://doi.org/10.1093/nar/gkw226

Shen, X., Jiang, C., Wen, Y., Li, C., & Lu, Q. (2022). A Brief Review on Deep Learning Applications in Genomic Studies. *Frontiers in Systems Biology*, *2*. https://doi.org/10.3389/fsysb.2022.877717